Unsupervised Text Embedding Space Generation Using Generative Adversarial Networks for Text Synthesis

Jun-Min Lee, Korea Advanced Institute of Science and Technology, I-BRICKS, ljm56897@gmail.com
Tae-Bin Ha, I-BRICKS, taebinalive@gmail.com

Abstract Generative Adversarial Network (GAN) is a data synthesis model that creates plausible data through the competition between a generator and a discriminator. Although GAN has been extensively studied for image synthesis, it has inherent limitations when applied to natural language generation. This is because natural language is composed of discrete tokens, and the generator faces challenges in updating its gradient through backpropagation. Therefore, most text-GAN studies generate sentences starting with a random token (or prompt) based on a reward system. Thus, the generators of previous studies are pre-trained in an autoregressive manner before adversarial training, resulting in data memorization where synthesized sentences reproduce the training data. In this paper, we synthesize sentences using a framework similar to the original GAN. More specifically, we propose Text Embedding Space Generative Adversarial Networks (TESGAN), which generate continuous text embedding spaces instead of discrete tokens to address the gradient backpropagation problem. Furthermore, TESGAN conducts unsupervised learning that does not directly refer to the text of the training data to overcome the data memorization issue. Also, TESGAN enables unconditional text synthesis during the inference phase by using random noise instead of tokens or prompts for text synthesis. By adopting this novel method, TESGAN can synthesize new sentences, demonstrating the potential of unsupervised learning for text synthesis. We look forward to extended research that combines large-scale language models with a new perspective on viewing text as continuous spaces.

1 Introduction

Generative Adversarial Network (GAN), as proposed by Goodfellow et al. (2014), is a popular model for data synthesis. GAN is an unconditional data generation algorithm that aims to generate plausible data in an unsupervised manner by fostering competition between a generator and a discriminator to capture the real data distribution. When GAN was initially introduced, it primarily focused on image synthesis, and extensive research was conducted to achieve high-quality synthetic data results (Arjovsky et al., 2017; Radford et al., 2015; Karras et al., 2018, 2021). Furthermore, GAN is commonly employed in the field of computer vision for data augmentation through image synthesis (Sandfort et al., 2019; Bowles et al., 2018; Antoniou et al., 2018; Tran et al., 2021). The GAN generator learns implicit density based on the discriminator's loss without direct reference to the training data. Consequently, GAN can prevent data memorization, where the model reproduces the training data. Additionally, GAN can synthesize various data by using random noise instead of a specific starting point, such as a designated start token.

Similar to images, unconditional text generation can function as a data augmentation technique by generating new text that resembles a given dataset. It also has practical applications, such as creating new documents by generating fictitious text information suitable for direct use. Consequently, several studies have attempted to apply GAN to natural language, but they have encountered limitations in natural language generation. The challenge arises from the fact that natural language is composed of discrete tokens, making it challenging for the GAN generator to directly update gradients through backpropagation. The gradient backpropagation issue in text-based GANs was first discussed by Yu et al. (2017), and numerous subsequent text-GAN research efforts aimed to address this problem using gradient policy-based reinforcement learning with a reward system. Furthermore, the previous text-GAN approaches necessitated pre-training the generator with supervised learning (autoregressive) before adversarial training due to convergence issue with the generator (Yu et al., 2017). Accordingly, we discovered that the generators of previous text-GAN approaches reproduce the training data (leading to data memorization) during text synthesis due to the autoregressive-based pre-training process, which becomes a significant

issue in generative models.

This paper introduces a novel framework known as Text Embedding Space Generative Adversarial Networks (TESGAN)[1], which enables backpropagation and prevents data memorization. TESGAN does not rely on a supervised, pre-trained autoregressive-based generator that generates discrete tokens for text synthesis. Our generator generates continuous text embedding spaces for text synthesis instead of discrete tokens, allowing training with gradient backpropagation. Furthermore, the fact that TESGAN deals with continuous spaces makes it possible for TESGAN's generator to be trained within the original GAN framework to mimic the real text embedding space. Moreover, TESGAN enables unconditional text generation, as it does not require the selection of a starting token (or prompt) for text synthesis. Our seed interpretation model then synthesizes sentences by interpreting the imitated continuous text embedding space created by the generator. During sentence synthesis, data memorization does not occur because TESGAN does not directly refer to the training text data but only learns from the continuous text embedding space. We use two datasets to conduct performance evaluations and general applicability experiments based on synthetic text generated by TESGAN. To assess the quality and diversity of synthesized text, we employ evaluation metrics such as Fréchet BERT Distance (FBD), Multi-sets-Jaccard (MSJ) (Alihosseini et al., 2019), Language Model score (LM) (de Masson d'Autume et al., 2019; Caccia et al., 2020), and Self-BLEU (SBL) (Zhu et al., 2018). In addition, we conducted human evaluations, and TESGAN achieved the highest average score. Lastly, we calculate the data memorization ratio and present the synthesized sentences to assess the potential of unsupervised learning and continuous embedding spaces for text synthesis.

2 Related Works

The most common method of text generation is to use an autoregressive-based language model via teacher forcing (Williams and Zipser, 1989). For example, extensive studies have been conducted on models using recurrent neural network (RNN) with Long Short-Term Memory (LSTM) cells (Hochreiter and Schmidhuber, 1997). Using LSTM, Graves (2013) successfully generated handwriting by predicting sequences, and Wen et al. (2015) synthesized sentences under specific conditions. Bowman et al. (2016) generated text after learning text embedding spaces with an autoregressive-based LSTM model and a variational autoencoder (VAE) architecture (Kingma and Welling, 2014). Policy Gradient with BLEU (PG-BLEU) calculates the BLEU (Papineni et al., 2002) score of synthesized sentences and takes them as a reward when updating the generator using policy gradient.

Numerous investigations have been conducted to utilize GANs for text synthesis. Sequence GAN (SeqGAN) (Yu et al., 2017) attempted to address the backpropagation problem by employing gradient policy-based reinforcement learning with a reward system. However, SeqGAN faced a reward sparsity issue, leading Lin et al. (2017) to introduce RankGAN, which replaced the previous regression-based discriminator with a novel ranker. RankGAN trains the discriminator to assign higher scores to more realistic sentences. MaskGAN (Fedus et al., 2018) utilized an LSTM-based generator to fill in masked parts of sentences with tokens during training. Since MaskGAN uses discrete tokens, gradient backpropagation is not possible for the generator. To overcome this challenge, the authors employed the actor-critic method, using the probabilities of candidate tokens from the discriminator as rewards during training. Che et al. (2017) proposed Maximum Likelihood Augmented Discrete GAN (Mali-GAN), which synthesizes text by minimizing Kullback-Leibler divergence (Kullback and Leibler, 1951). Leak-GAN (Guo et al., 2018) alleviated issues related to sparseness and the lack of intermediate information by providing leaked information from the discriminator.

Several studies have aimed to address the gradient backpropagation problem without relying on reward-based reinforcement learning. TextGAN (Zhang et al., 2017) introduced kernel-based moment-matching, which enforces empirical distributions of real and synthetic text by using LSTM and Convolutional Neural Networks (CNN) for the generator and the discriminator, respectively. Feature Mover GAN (FM-GAN) (Chen et al., 2018) defined the feature-mover's distance (FMD) and learned it by minimizing the FMD between real and fake sentences. Both TextGAN and FM-GAN utilized LSTM generators that generate discrete tokens using the *soft-argmax* trick instead of relying on reinforcement learning. Relational GAN (RelGAN) (Nie et al., 2019) applied relational recurrent neural networks (Santoro et al., 2018) and attempted to address the gradient backpropagation issue using Gumbel-softmax (Jang et al., 2017). However, since these approaches employed autoregressive (e.g., LSTM) generators, they explicitly referenced the training text data during model training. Consequently, previous studies faced challenges in avoiding complete data memorization while synthesizing sentences due to an autoregressive generator. Lastly, Transformer-based Implicit Latent GAN (TILGAN) (Diao et al., 2021) adopted a similar approach to TESGAN for addressing the gradient backpropagation issue based on the

[1] https://github.com/ljm565/TESGAN

embedding space. However, TILGAN differs from TESGAN in that it was trained on a latent space compressed by the encoder, configured as an autoencoder transformer, and did not utilize embeddings learned from real language models.

Most of the aforementioned text-GAN models require the first token or prompt to synthesize text due to their autoregressive generators. TESGAN stands apart from these models as it generates text embedding space directly from random noise, eliminating the need for selecting tokens. Our TESGAN is the first text-GAN model that learns the real text embedding space without relying on an autoregressive generator.

3 Text Embedding Space GAN

TESGAN aims to generate the seeds required for synthesizing plausible text. These generated seeds (fake seeds) from the generator, along with the real seeds from the real text, are passed to the discriminators for training within the GAN framework. Once the training of TESGAN is complete, the pre-trained seed interpretation model synthesizes text using the fake seed created by the generator.

3.1 Seed for Text Synthesis

We denote a text sequence as $S = w_1, \ldots, w_T$ (T is the sequence length). An autoregressive-based language model calculates the probability of the text sequence S as a product of conditional probabilities. If we assume that S is a complete sentence, then the sequence $D = S_1, \ldots, S_N$ (N is the dialogue length) can be viewed as multi-turn sentences. Let $S_1 = w_1, \ldots, w_m$ and $S_2, \ldots, S_N = w_{m+1}, \ldots, w_M$ denote the first sentence and the subsequent sentences, respectively (M is the total length of the multi-turn sentences D). The subsequent sentences after the first sentence can be predicted from a product of conditional probabilities:

$$p(S_2, \ldots, S_N | S_1) = \prod_{i=m+1}^{M} p(w_i | w_1, \ldots, w_{i-1}) \tag{1}$$

In other words, the first sentence can generate subsequent text using an autoregressive-based language model trained with multi-turn sentences. Therefore, the first sentence can serve as a seed. Meanwhile, the generator of TESGAN generates the first sentence as a continuous embedding space instead of discrete tokens to enable gradient backpropagation. Consequently, a continuous embedding space of the first sentence is defined as a seed.

3.2 Seed Interpretation Model

We define a seed in Section 3.1, and the seed interpretation model $f_\theta(\cdot)$ is used to synthesize text based on the seed. To synthesize text, the seed interpretation model must first be trained with multi-turn sentences in an autoregressive manner, similar to general language modeling, before adversarial training, as shown in Figure 1 (left), with the following loss function:

$$\mathcal{L}_{LM} = -\frac{1}{N} \sum_{n=1}^{N} \log \frac{exp(x_{n,y_n})}{\sum_{c=1}^{C} exp(x_{n,c})} \tag{2}$$

This enables the generator to synthesize appropriate text by utilizing the fake embedding space it creates during the inference phase. More detailed explanations will be provided in the following section. As a result, the model has to be trained on data consisting of multi-turn sentences $D = S_1, \ldots, S_N$, where each sentence has a maximum length of L, meaning the total number of tokens in D is $N \times L$. When constructing the multi-turn sentence data, the special token $[CLS]$ is inserted only at the beginning of the first sentence, and each sentence is distinguished by adding the special token $[SEP]$ at the end. If the length of the tokenized sentence is less than L, the sentence is padded with the special token $[PAD]$:

$$\begin{gathered} S_1 = w_1^1, \ldots, w_{|S_1|}^1, \ldots, w_L^1 \\ (w_1^1 = [CLS], w_{|S_1|}^1 = [SEP], w_{|l>S_1|}^1 = [PAD]) \end{gathered} \tag{3}$$

$$\begin{gathered} S_{i(i>1)} = w_1^i, \ldots, w_{|S_i|}^i, \ldots, w_L^i \\ (w_{|S_i|}^i = [SEP], w_{|l>S_i|}^i = [PAD]) \end{gathered} \tag{4}$$

where S_1 and S_i represent a seed sentence and subsequent text. Let H_{real} denote the real seeds from TESGAN. As shown in Figure 1, the real seed is an embedding space of a sentence obtained by applying the sum of the token embedding and the positional embedding to the sigmoid function. Since most sentences can exist before others as long as the seed interpretation model is trained with multi-turn sentences, a significant portion of them can be used as seeds for text generation. Therefore, most of the sentences can be used as real seeds:

$$\begin{aligned} H_{real} &= \sigma\big(W_{emb}(S_1) + W_{pos}(S_1)\big) \\ &\approx \sigma\big(W_{emb}(S_n) + W_{pos}(S_n)\big) \in \mathbb{R}^{L \times d} \end{aligned} \tag{5}$$

where L and d represent sequence length and embedding dimensions. H_{real} from the real text can be viewed as continuous spaces, similar to images, and the well-pretrained seed interpretation model can predict the next sentence S_{n+1} properly as illustrated in Figure 1 (right):

$$S_{n+1} = f_\theta\Big(\sigma\big(W_{emb}(S_n) + W_{pos}(S_n)\big)\Big) = f_\theta(H_{real}) \in \mathbb{Z}^L \tag{6}$$

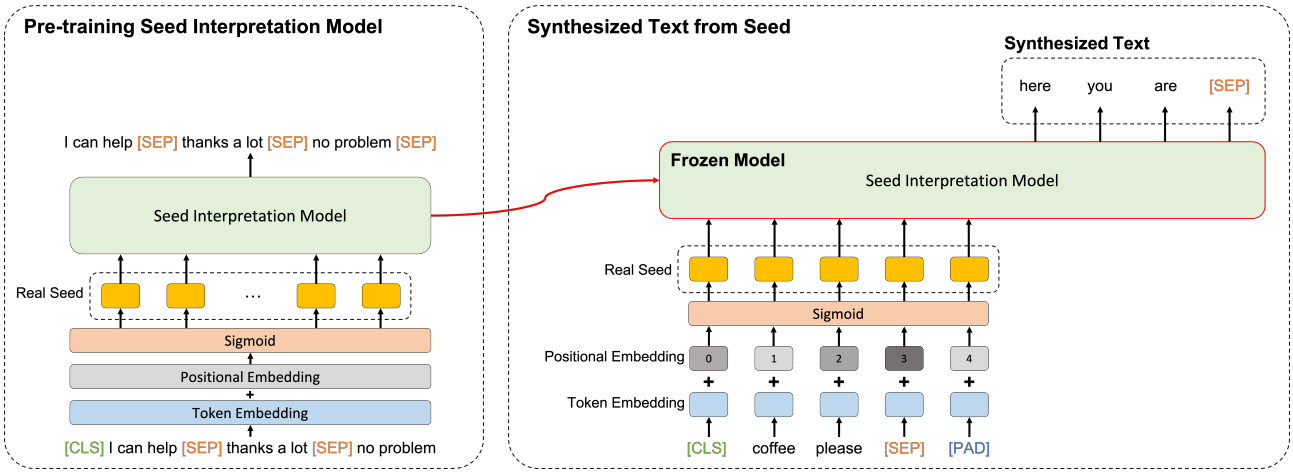


Figure 1: Illustration of the seed interpretation model. The seed interpretation model is pre-trained with multi-turn sentences before adversarial training (left). After pre-training, the model's parameters are frozen, allowing it to synthesize text from the seed. The right figure implies that text can be synthesized from the seed. The $[PAD]$ tokens following the $[SEP]$ tokens are omitted in the left part for clarity.

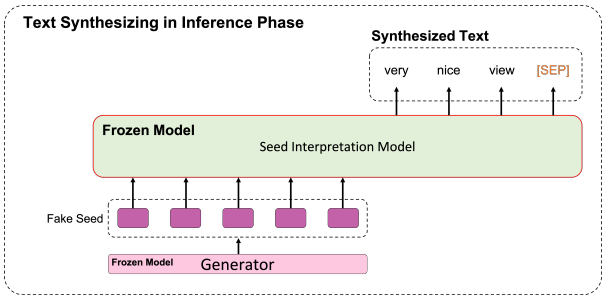


Figure 2: Illustration of text synthesizing method using the seed interpretation model in the inference phase.

As a result, text synthesis is carried out as the seed passes through the seed interpretation model to predict the subsequent sentence.

Applying to Unconditional Text Synthesis

Here, we assume that the training of the TESGAN framework, including adversarial training, is fully completed and describe how the seed interpretation model synthesizes text during the inference phase. Let $f^*_\theta(\cdot)$ and $g^*_\phi(\cdot)$ denote the frozen seed interpretation model and frozen generator, respectively. We can now synthesize text during the inference stage using the well-trained $f^*_\theta(\cdot)$ and $g^*_\phi(\cdot)$. At this point, $g^*_\phi(\cdot)$ will generate a fake seed H_{fake} with the same dimensions as H_{real}, as shown in Equation 7. Then, $f^*_\theta(\cdot)$ can synthesize text using the fake seed, as shown in Figure 2. In other words, if the generator can skillfully create fake seeds H_{fake} that imitate the distributions of H_{real}, then H_{fake} can also generate appropriate subsequent sentences (a.k.a synthetic text). However, no matter how excellently H_{fake} is generated by the generator, it is useless if it cannot be interpreted; therefore, training the seed interpretation model is crucial. We use the pre-trained GPT-2 (Radford et al., 2018)[2] model and fine-tune it with multi-turn text data to serve as the seed interpretation model. In addition, this model is only used to provide H_{real} from the real text with frozen parameters during adversarial training. Thus, the seed interpretation model never affects the training of the generator and the discriminator during adversarial training, and vice versa. More detailed specifications of the seed interpretation model are explained in Appendix A.

Synthesizing text based on the generated fake seeds H_{fake} by the generator is entirely different from autoregressive prompting. This is because prompting methods (Wei et al., 2021; Ouyang et al., 2022; Chung et al., 2022) function by providing discrete tokens as input to a model that generates the next tokens based on the previous one. On the other hand, the generator of TESGAN creates continuous spaces H_{fake} for synthesizing text from random noise, enabling unconditional text synthesis without explicit human instruction. Furthermore, research is actively being conducted to leverage continuous spaces (learnable query) for flexible model training (Lester et al., 2021; Alayrac et al., 2022; Li et al., 2023; Dai et al., 2023). Most research explores various methods, including using a fixed learned query after model training or memorizing multiple learned queries and selecting them selectively as needed in different situations. However, the TESGAN framework differs from the mentioned studies in that its primary objective is to generate appropriate queries through the gernerator to

[2] https://huggingface.co/docs/transformers/model_doc/gpt2

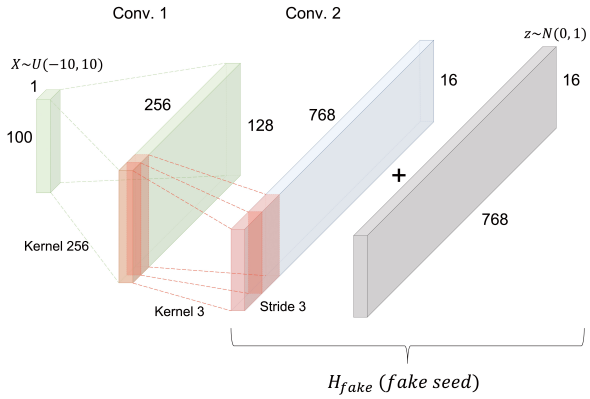


Figure 3: Illustration of the generator. P-TESGAN makes perturbed seeds by adding zero-centered normal distribution noise z (gray) to the output (blue) from the generator.

produce appropriate sentences.

3.3 Generator

Both real and fake seeds (H_{real} and H_{fake}) are essential for adversarial training. Real seeds can be obtained from the real text, as described in the seed interpretation model, and fake seeds are generated by the generator. In most text-GAN models reported so far, fake sentences are obtained from a text-based pre-trained autoregressive generator. Consequently, data memorization occurs, where several synthetic sentences reproduce the training data. To prevent data memorization, our generator does not use a pre-trained autoregressive-based model and does not explicitly reference the text in the training data during adversarial training. Our generator aims to create suitable fake text embedding spaces H_{fake} in an unsupervised manner (GAN framework) by referencing real text continuous spaces H_{real}.

As shown in Figure 3, the generator $g_\phi(\cdot)$ is composed of two convolutional layers and generates seeds from the uniform distribution noise X within an interval of $[-10, 10)$ to create diverse forms of the seeds. Additionally, using random noise has the advantage of not having to select the first token in the text synthesis process after model learning. The final H_{fake} can be obtained by Equation 7:

$$H_{fake} = g_\phi\big(X \sim U(-10, 10)\big) \in \mathbb{R}^{L \times d} \tag{7}$$

where L and d represent sequence length and embedding dimensions. As a result, the embedding space created by the generator has the same dimension as the real seed. Moreover, we also compare an additional model, perturbed TESGAN (P-TESGAN). P-TESGAN creates perturbed seeds by adding zero-centered normal distribution noise z to the generator output. P-TESGAN is expected to learn more robustly by perturbating the generator output. Please refer to Appendix A for detailed model information.

3.4 Objective Functions

Since the generator does not refer to text during adversarial training, its performance is determined by the loss of the discriminator. Thus, we propose four types of loss to update the parameters of the generator and the discriminator.

3.4.1 Discriminators

Sentence structure is important for constructing a complete sentence. Since the real seeds are made from perfect sentences, they maintain structural representations of sentences. Therefore, the fake seeds should capture the structural features of the real seeds. As shown in Figure 4a, we use Bidirectional Encoder Representations from Transformer (BERT) (Devlin et al., 2019) $d_\alpha(\cdot)$ called Seed Structure Discriminator (SSD) to capture the structural features of sentences, and the first hidden state is used to predict whether the seed is real or fake.

The order of tokens is also important for constructing sentences. It is possible to predict whether a sentence's order representation of a seed is correct because both real and fake seeds have a dimension of (***sequence length*** * *embedding dimensions*). To do this, as shown in Figure 4b, we use Bidirectional LSTM $d_\beta(\cdot)$ called Seed Order Discriminator (SOD) to consider both forward and backward directions of sentences. The concatenation of the first and the last hidden states is used to predict whether the seed is real or fake.

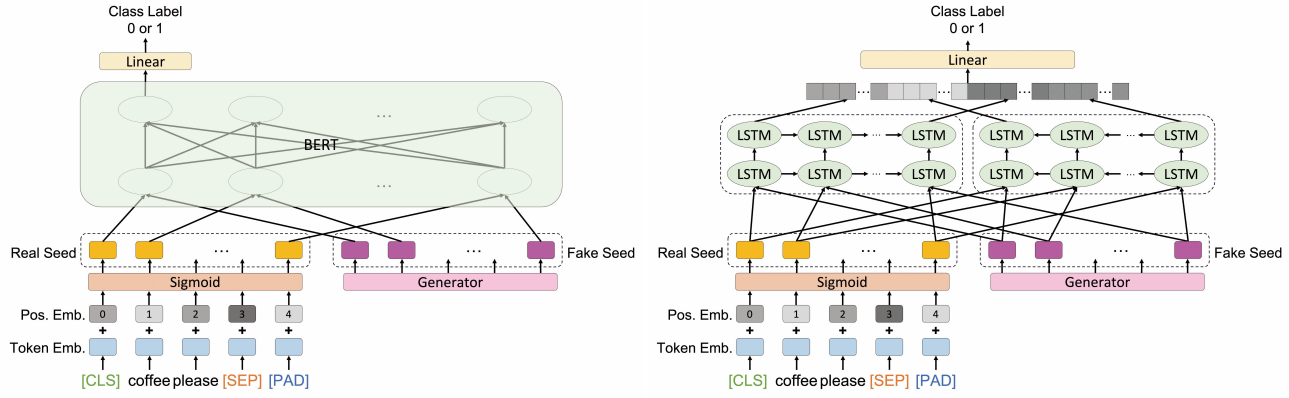
During adversarial training, both discriminators are trained to predict whether the seeds are real (label 1) or fake (label 0), while the generator is trained to fool the discriminators by predicting fake seeds as 1. The loss function of the discriminators is defined by the following equation, which updates both the discriminators and the generator:

$$\mathcal{L}_D = -\frac{1}{N}\sum_{i=1}^{N}\big[y_i \log x_i + (1-y_i)\log(1-x_i)\big] \tag{8}$$
$$x = predicted, y = target$$

Additional information regarding the size and descriptions of the discriminators can be found in Appendix A.

3.4.2 Generator Helpers

During adversarial training, it is challenging for the generator to learn solely from the discriminators introduced in Section 3.4.1. Therefore, in this section, two



(a) Seed Structure Discriminator (SSD)

(b) Seed Order Discriminator (SOD)

Figure 4: Illustrations of the two discriminators. SSD predicts whether the seed is real or fake using the $[CLS]$ special token's feature. SOD considers both forward and backward contexts of the seed.

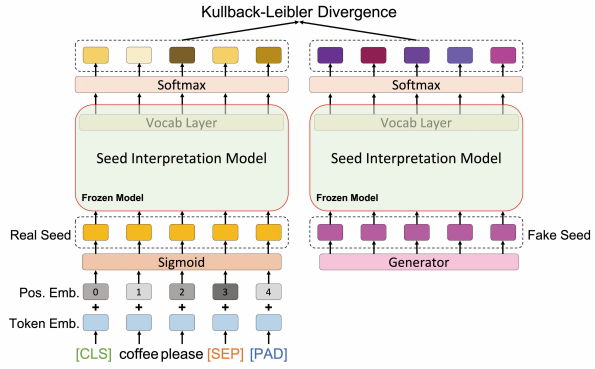


Figure 5: Illustration of Seed Distribution Prediction (SDP). SDP is used to enhance the fake seeds of the generator during adversarial training by minimizing the distance between real and fake seed distributions.

auxiliary tasks are introduced to aid the training of the generator.

Capturing the distribution of the text embedding space is important, and for this purpose, we employ Seed Distribution Prediction (SDP). However, since the text embedding space cannot be directly used as a probability distribution, the output of the seed interpretation model is utilized. Specifically, when a seed passes through the frozen seed interpretation model, the output dimension of (*sequence length * vocabulary size*) is obtained through the softmax function, which can be used as a probability distribution. The loss is calculated using the Kullback-Leibler divergence between the distributions of the real and the fake seeds. SDP is used solely for updating the generator during adversarial training:

$$\mathcal{L}_{SDP} = \sigma(f^*_\theta(H_{real})) \log \frac{\sigma(f^*_\theta(H_{real}))}{\sigma(f^*_\theta(H_{fake}))} \tag{9}$$

where the σ and $f^*_\theta(\cdot)$ mean *softmax* function and the frozen seed interpretation model. More detailed figure of SDP is illustrated in Figure 5.

The sentences used as the seeds are composed of tokens explained in Equation 3. Additionally, we apply Seed Frame Prediction (SFP) since the structures of seeds are somewhat formalized. Therefore, we calculate the Mean Absolute Error (MAE) and Mean Squared Error (MSE) to make the form of a fake seed similar to a real one. If we train the fake seeds using MAE and MSE, the fake seeds from the generator can become blurred. However, the loss of SFP is relatively small compared to that of SSD, SOD, and SDP; therefore, SFP does not adversely affect the generator. SFP is used only for updating the generator during adversarial training:

$$\begin{aligned} \mathcal{L}_{SFP} &= \|\mu_r - \mu_f\|_2^2 + \|H_{real} - H_{fake}\|_1 \\ \mu_r &= avg(H_{real}), \mu_f = avg(H_{fake}) \end{aligned} \tag{10}$$

The full loss function, including the seed interpretation model's loss, is described in Appendix B.

4 Text Synthesis Experiments

4.1 Dataset

In this experiment, we use two datasets consisting of multi-turn sentences to train the seed interpretation model and perform the text synthesis experiment.
DailyDialog[3] (Li et al., 2017) is multi-turn conversation data used for training open-domain dialogue generation models. It consists of chit-chat-style multi-turn English conversations, and we select this data for domain-independent text synthesis. This dataset is used to evaluate the performance of TESGAN and other baselines.
IMDb[4] (Maas et al., 2011) contains highly polar movie

[3] http://yanran.li/dailydialog
[4] https://huggingface.co/datasets/imdb

Algorithm 1 Text Embedding Space GAN

Require: Seed interpretation model f_θ; Generator g_ϕ; BERT discriminator d_α; LSTM discriminator d_β; Multi-turn data $D = \{S_{1:N}\}$; Sentence data $S_i = \{w^i_{1:L}\}$.

```
Pre-train f_θ using D.
Initialize g_φ, d_α, d_β with random weights
  φ, α, β ~ N(0, 0.08).
Freeze the f_θ.
while TESGAN converges do
    for d-steps (during odd epoch) do
        Get real data from f_θ using S with positive label 1.
        Make fake data from g_φ with negative label 0.
        Update α and β via results of d_α and d_β.
    end for
    for g-steps do
        Make fake data from g_φ with positive label 1.
        Calculate SDP and SFP.
        Update φ via results of d_α, d_β, SDP and SFP.
    end for
end while
```

reviews and is widely used for sentiment classification tasks. Each human-written movie review consists of several sentences, and we used this data as multi-turn data. This dataset is rougher and has a larger volume compared to DailyDialog. We evaluate the general applicability by synthesizing sentences based on IMDb-trained TESGAN. Statistics of the two datasets are shown in Appendix C.

4.2 Training Steps

TESGAN training has two steps. First, the seed interpretation model must be pre-trained with multi-turn data to interpret the seeds. In the performance and general applicability experiments, we train the model on the 11k and 25k multi-turn sets of DailyDialog and IMDb, respectively, as shown in Table 7. Then, the model that achieves the highest BLEU-4 score in the validation set is selected in each experiment.

The second step is adversarial training. After pre-training the seed interpretation model, the generator and the discriminator learn through adversarial training. For adversarial training, real and fake seeds are created by the embedding part of the frozen seed interpretation model and the generator, respectively. Since real seeds can be generated from a significant number of sentences, all 87k and 300k sentences in each training set used in the experiment mentioned above are used to create the real seeds via Equation 3. We also generate the same number of fake seeds as real seeds for adversarial training, and the following equation represents what the discriminator and generator aim to optimize during adversarial training:

$$\begin{aligned} \mathcal{D}_{\mathcal{L}} &= \max_{\alpha,\beta} \mathbb{E}_{x\sim H_{real}}\Big[\log d_{\alpha,\beta}(x)\Big] \\ \mathcal{G}_{\mathcal{L}} &= \max_{\phi} \mathbb{E}_{z}\Big[\log d_{\alpha,\beta}(g_\phi(z))\Big] + \mathcal{L}_{SDP} + \mathcal{L}_{SFP} \end{aligned} \tag{11}$$

where $d_{\alpha,\beta}$ means SSD, SOD respectively. $\mathcal{D}_{\mathcal{L}}$ implies updating the parameters of the discriminator to accurately predict real seeds as 1 from the perspective of real seeds. $\mathcal{G}_{\mathcal{L}}$ also means updating the generator so that the discriminator predicts the fake seeds created by the generator as 1. This approach helps partially resolve the learning imbalance problem between the generator and the discriminator (Goodfellow et al., 2014). Further discussion of the above pseudocode and optimization methods is covered in Section 6.1. Lastly, hyperparameters and experiment setup are described in Appendix D.

4.3 Evaluation Metric

Target-oriented evaluation metrics, such as BLEU and ROUGE (Lin, 2004), are not suitable for evaluating synthetic text. This is because each synthesized sentence from random noise has no corresponding target, and the generative models aim to synthesize plausible data based on real data distribution without copying the training data. Therefore, we employ several metrics that can evaluate unconditional text generation.

4.3.1 Fréchet BERT Distance (FBD)

de Masson d'Autume et al. (2019) proposed Fréchet Embedding Distance (FED) to evaluate the quality of synthetic text, inspired by Fréchet Inception Distance (FID) (Heusel et al., 2017). Alihosseini et al. (2019) proposed FBD, an improved version of FED, to measure the quality and diversity of synthesized text using a pre-trained BERT. The features of real and synthesized text obtained by the pre-trained BERT are assumed to have Gaussian distributions, and FBD is the distance between them:

$$FBD = \sqrt{\|\mu_r - \mu_f\|_2^2 + tr\big(\Sigma_r + \Sigma_f - 2(\Sigma_r\Sigma_f)^{0.5}\big)} \tag{12}$$

where μ and Σ show the mean vectors and the covariance matrices of the real and fake seed features.

4.3.2 Multi-Sets-Jaccard (MSJ)

Each synthesized sentence has no corresponding target; thus, we select MSJ (Alihosseini et al., 2019), which calculates the score between real and synthesized text sets. The Jaccard Index determines the similarity of two sets, calculating the ratio of the cardinality of their intersection to that of their union. Inspired by the Jaccard

Index, MSJ focuses on the similarity of the n-gram frequencies of text in the two sets, s_r and s_f, which are the real and synthesized text sets, respectively:

$$MSJ_n = \frac{\sum_{g \in G_n} \min\left(C_n(g, s_r), C_n(g, s_f)\right)}{\sum_{g \in G_n} \max\left(C_n(g, s_r), C_n(g, s_f)\right)} \tag{13}$$

where G_n and $C_n(g, s)$ mean the n-gram in $s_r \cup s_f$ and the normalized counts of the n-gram in set s. Additionally, this n-gram-based synthesized sentence evaluation method is a common approach in the field of unconditional text generation (Yu et al., 2017; Press et al., 2017; Fedus et al., 2018).

4.3.3 Language Model score (LM)

de Masson d'Autume et al. (2019); Caccia et al. (2020) proposed LM, which can evaluate the quality of generated samples using a well-trained language model. LM measures the quality of generated samples, meaning that scores of the bad samples are poor under a well-trained language model. We select the pre-trained GPT-2[2] as a well-trained language model. LM is calculated as the cross-entropy results between the output and input of GPT-2.

4.3.4 Data Synthesis Ratio (DSR)

DSR considers not only the data memorization ratio between the training and synthesized data but also synthetic diversity itself. Short sentences identical to training data, such as "*I'm fine*", can be synthesized by coincidence. Therefore, sentences longer than two-thirds of the maximum sentence length that perfectly reproduce the training data are considered memorized data. Considering these conditions, we can calculate DSR using the following equation:

$$\begin{gathered} R_{syn} = \frac{|S_{syn} - S_{train}|}{|S_{syn}|}, R_{unq} = \frac{|S_{unq}|}{|S_{syn}|} \\ DSR = \frac{2 * R_{syn} * R_{unq}}{R_{syn} + R_{unq}} \end{gathered} \tag{14}$$

where S_{syn} and S_{train} indicate synthesized and training text set respectively. S_{unq} means the set of unique sentences of synthesized text results. If the synthesized sentences in S_{syn} do not reproduce any of the sentences in S_{train}, R_{syn} would be 1. Similarly, if the synthesized text in S_{syn} is all unique, R_{unq} will be 1. The final DSR is calculated as the harmonic mean of R_{syn} and R_{unq} ratios.

4.3.5 Self-BLEU (SBL)

Zhu et al. (2018) first proposed SBL to measure diversity of token combination. The original BLEU evaluates the degree of n-gram overlap (similarity) between one hypothesis sentence and multiple reference sentences. However, unconditionally generated text does not have specific targets, so it is not suitable for BLEU evaluation. SBL is widely used to solve this problem. SBL can evaluate n-gram-level similarity by regarding one sentence as a hypothesis and the rest as references in a synthetic text set. Since SBL evaluates based on the generated text set itself, it is not able to evaluate the quality of the synthetic text, but it is possible to evaluate the diversity of token combinations based on n-gram. Additionally, the difference between SBL and DSR lies in their evaluation criteria. DSR assesses data memorization by comparing the generated text set with the training dataset, while also considering the diversity of not n-gram-based but generated complete sentences themselves.

4.4 Baselines

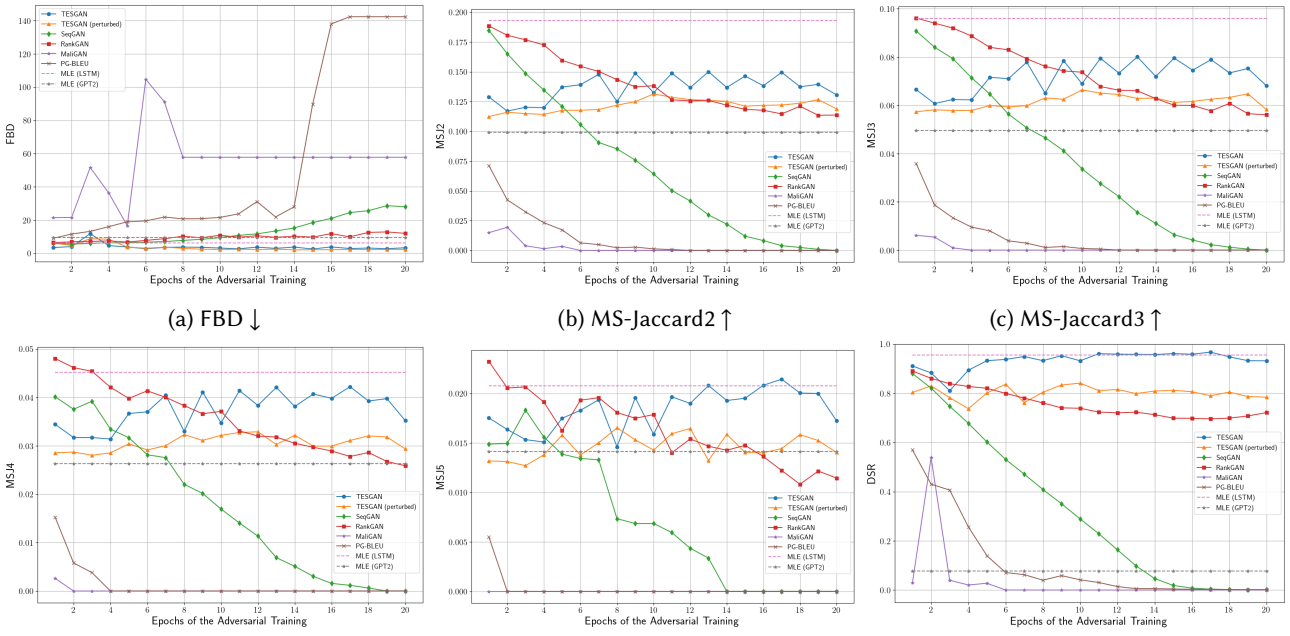
In this paper, we compare our two models with the following approaches: LSTM-based Maximum Likelihood Estimation (MLE-L), PG-BLEU, SeqGAN, RankGAN, and MaliGAN. MLE-L represents the pre-training result of the generator, which all the other models undergo before adversarial training. The pre-trained generators with the lowest loss in the validation set were chosen for each method, including MLE-L. We compare these models with our original TESGAN and P-TESGAN, which is trained by adding zero-centered normal distribution noise z to the generator's output. We also evaluate the GPT-2-based pre-trained seed interpretation model (MLE-G) used in the TESGAN framework. Since MLE-based models are trained without adversarial training, they are shown as baselines in Figure 6. Finally, to demonstrate that the results of the TESGAN-based models are not solely dependent on the seed interpretation model but rather on seeds created by the generator, we present the outcomes when using Gaussian random noise as input for the seed interpretation model.

5 Results

5.1 Metric-based Evaluation

In this section, we compare the results of the synthesized text at every epoch of adversarial training using the metrics mentioned in Section 4.3. This experiment was performed with models trained on the DailyDialog dataset. Since Fréchet BERT Distance (FBD) and Multi-Sets-Jaccard (MSJ) require a real text corpus, the test set is used as the real text corpus. Data Synthesis Ratio (DSR) is calculated with the training set as the data memorization ratio needs to be computed.

The FBDs of the TESGAN-based models are lower than the MLE-based autoregressive results, while the



(a) FBD ↓ (b) MS-Jaccard2 ↑ (c) MS-Jaccard3 ↑

(d) MS-Jaccard4 ↑ (e) MS-Jaccard5 ↑ (f) DSR ↑

Figure 6: Illustration showing the results of the text-GAN models. In previous research, adversarial training is conducted after the generator pre-training. MLE is represented as a baseline because it is a supervised pre-trained generator without adversarial training.

Method	FBD^{Q,D} ↓	MSJ4^{Q,D} ↑	MSJ5^{Q,D} ↑	DSR^{M} (R_{syn}, R_{unq}) ↑	LM$*^{Q}$ ↓	SBL3$*^{D}$↓	SBL4$*^{D}$ ↓
TESGAN (ours)	2.899	0.042	0.021	**0.967** (**1**, 0.936)	4.236	0.743	0.623
P-TESGAN (ours)	**2.274**	0.032	0.014	0.841 (0.997, 0.727)	3.642	0.790	0.702
SeqGAN (Yu et al., 2017)	6.153	0.040	0.015	0.880 (0.883, 0.877)	5.094	0.420	0.266
RankGAN (Lin et al., 2017)	6.409	**0.048**	**0.023**	0.890 (0.895, 0.886)	5.123	0.446	0.290
MaliGAN (Che et al., 2017)	21.436	0.003	0	0.030 (**1**, 0.015)	-	-	-
PG-BLEU (Yu et al., 2017)	9.002	0.015	0.006	0.569 (0.555, 0.584)	4.584	0.628	0.484
MLE-L (Yu et al., 2017)	6.284	0.045	0.021	0.955 (0.925, **0.987**)	5.168	**0.403**	**0.242**
MLE-G	9.592	0.026	0.014	0.078 (**1**, 0.040)	**3.543**	0.948	0.944
Random Noise †	14.142	0.016	0.006	0.930 (1, 0.869)	4.562	0.516	0.404

Table 1: Performance of the models. P-TESGAN denotes the perturbed TESGAN. † is the result of directly entering Gaussian random noise as an input to the seed interpretation model. The second group of models consists of autoregressive models. $*$ denotes a metric not considered when selecting the best model. - denotes that the confidence of the result is low because the quality of the synthesized sentence is poor. The superscript of each metric represents what each metric can measure (Q: quality, D: diversity, M: data memorization).

baselines increase after having the lowest value at the end of the first epoch of adversarial training, as shown in Figure 6a. In terms of MSJ, as shown in Figure 6b-6e, the previous studies report lower values than the TESGAN-based models at the end of adversarial training, despite having higher results in the beginning. On the other hand, MSJ results of the TESGAN-based models slightly increase during adversarial training. Moreover, some MSJ5 results of the original TESGAN are higher than MLE-L during adversarial training, as shown in Figure 6e. In the case of DSR, as shown in Figure 6f, the original TESGAN also increases during adversarial training, and some results are higher than MLE-L. On the other hand, the results of the previous studies decrease during adversarial training, resulting in lower values than the TESGAN-based models in the end. As adversarial learning progresses, the results of the baselines deteriorate because the LSTM generator tends to generate only a few unique sentences. Furthermore, we will explain the reasons why the MLE-G results of the GPT-2 base are relatively poor in the following section.

We chose the best model of each method considering the FBD, MSJ, and DSR results because these met-

TESGAN (17-epoch, DailyDialog)	P-TESGAN (10-epoch, DailyDialog)	Random Noise
I'm so glad you finally got on the train.	Hello, Mr. Smith. I'm Mary.	Anything I have called three weeks
I just lost my job.	I just want to tell you the truth.	Is Is Is Is Is Is Is
Yeah. You mean the network connection?	It's the end of the world.	Left and go to go to go to go
What happened?	What do you want to do in this company?	Mr Moon, Mr Moon…Mr Moon…
So you have to wait for a while.	He just broke up with Ann.	are you have finished 6 items?

TESGAN (18-epoch, IMDb)
This is probably one of the best of the best of the series.
I was bored to think about how stupid this movie was.
"The Deadly Loved One" is the story of a rebellious college basketball
I have to say, this is the worst film I have ever seen.
I was very excited to see it, anticipating Christmas eve.
This movie was one of the best of the year for me.

Table 2: Example of unconditionally synthesized sentences. P-TESGAN denotes the perturbed TESGAN.

rics evaluate quality, diversity, and data memorization. We evaluated the text generated by each model per epoch using the metrics mentioned above and compared the best-performing models[5]. According to Table 8 in Appendix E, we compare the baselines at 1-epoch with TESGAN and P-TESGAN at 17 and 10-epochs, respectively. After selecting the best models, we calculated the Language Model score (LM) and Self-BLEU (SBL) based on the text generated by each model. As shown in Table 1, the TESGAN-based models show the highest results in FBD and DSR. Also, the TESGAN-based models show comparable results in MSJ compared to the baselines and display the highest results among the adversarial-based methods in terms of LM score. However, in terms of SBL, TESGAN-based models perform worse than the baselines. In addition, the results of Gaussian random noise demonstrate that the TESGAN results are attributed to the seeds from the generator. The first group of Table 2 shows the synthetic text by TESGAN and Gaussian random noise. In conclusion, MLE-L is a supervised pre-trained generator applied before adversarial training, but most of the result curves of the prior methods showed lower performance than MLE-L during adversarial training. On the other hand, our TESGAN-based models showed better results than MLE-L or improved performance during adversarial training. Finally, the results according to the epoch of the LM and SBL of each model are shown in Appendix E.

5.2 Analysis of Autoregressive Models

In this section, we will analyze the results of the MLE-based autoregressive models. Other baseline models pretrain an LSTM-based generator before starting adversarial training, while the TESGAN framework employs a GPT-2-based pre-trained seed interpretation model. The results of the MLE-based models in Section 5.1 are based on the evaluation of corpora generated in an autoregressive manner using the two pre-trained models. MLE-based models generate sentences in an autoregressive manner, starting from a specific $[start\_token]$ and predicting the next token. If the model predicts the next token in a greedy manner, all generated sentences would be identical, exhibiting deterministic behavior. To prevent this, MLE-based models sample the next token based on the probability logits (Yu et al., 2017). This way, MLE-L results in diverse token choices since the logit probability differences are not large. On the other hand, MLE-G training fits the data better than LSTM-based models, resulting in significantly larger differences in the logits of the next token. As a consequence, MLE-G is relatively deterministic compared to MLE-L. Therefore, when generating sentences without the use of the softmax temperature technique (Hinton et al., 2015), MLE-G delivers high quality, but it struggles to produce a variety of sentences. In practice, sentences generated by MLE-G lack diversity, which led to relatively lower results in Section 5.1. However, it is worth noting that while diversity may be lacking, the quality of the generated sentences is high, and this aspect will be demonstrated in the next section.

5.3 Human Evaluation

We conducted human evaluations based on the corpora generated by each model. The corpora, comprising 50 randomly selected unique sentences that do not duplicate those from the training set, were assessed by 10 annotators. We asked annotators to give higher scores to corpora that contained more natural sentences on a scale from 1 to 5. The scores presented in Table 3 represent the average scores assessed by each person. According to Table 3, TESGAN received the highest score, with MLE-G achieving the second-highest result. As mentioned in Section 5.2, MLE-G, despite facing challenges in generating diverse sentences, was able to pro-

[5]Detailed results are shown in Appendix E

Method	Avg. Score
TESGAN (ours)	**4.2**
P-TESGAN (ours)	3.4
SeqGAN (Yu et al., 2017)	2.4
RankGAN (Lin et al., 2017)	2.0
MaliGAN (Che et al., 2017)	1.0
PG-BLEU (Yu et al., 2017)	1.6
MLE-L (Yu et al., 2017)	3.0
MLE-G	3.8

Table 3: Human evaluation scores ($1 \sim 5$).

duce high-quality sentences.

5.4 General Applicability

In this section, we trained TESGAN with IMDb using a larger volume than DailyDialog to assess the general applicability of TESGAN. The IMDb-trained TESGAN is evaluated with both the DailyDialog test set (zero-shot) and the IMDb test set (non-zero-shot). Figures 7a and 7b display the zero-shot and non-zero-shot test results of the IMDb-trained TESGAN, respectively. Additionally, the zero-shot results generally exhibit a similar trend to the non-zero-shot test, suggesting that the model is being trained without bias toward the training data. Furthermore, the second group in Table 2 presents the synthetic text results of the IMDb-trained TESGAN. Both the zero-shot and text synthesis results indicate that TESGAN's outcomes do not vary significantly depending on the dataset, implying that TESGAN generalizes well and can be trained on diverse datasets. Figure 7c also illustrates the DSR, LM, and SBL results of the IMDb-trained TESGAN. Since these metrics are evaluated not on the test set but on generated text data, they consistently yield results regardless of the zero-shot test.

5.5 Error Analysis

We also conducted three additional TESGAN trainings without setting a manual seed in the code to confirm reproducibility. To assess whether the sentences generated by the model for each trial converge as adversarial training progresses, we calculated the Standard Error of the Mean (SEM) based on the average results. SEM is equivalent to the standard deviation of a sample mean taken from a population and represents the standard deviation that indicates the extent of variability in sample means. SEM is calculated by $\frac{\sigma}{\sqrt{n}}$ (σ and n denote average results and the number of trials). As a result, the overall tendency of training outcomes during adversarial training is similar. Furthermore, the SEM of each epoch decreases during adversarial training, indicating that each TESGAN converges. Figure 8 displays the results of the four experiments, including the average results and SEM.

6 Discussion

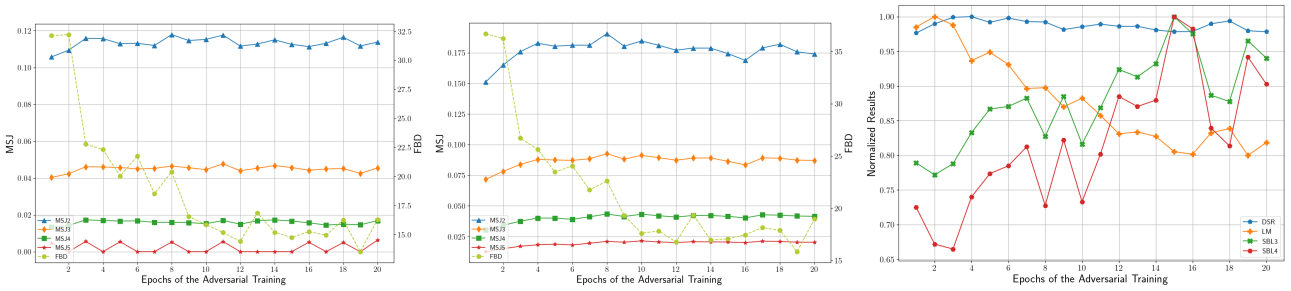
6.1 Generator and Training Strategy

We found that the performance of the TESGAN framework depends on the generator's architecture. When ReLU was used, dying ReLU (Lu et al., 2019) occurred, where the negative values became zero, making it unsuitable for text synthesis where diversity is important. Additionally, the hyperbolic tangent (tanh) was not adequate due to the problem of gradient vanishing (Wang et al., 2019). Consequently, we adopted Leaky ReLU (Maas et al., 2013) as the activation function between two convolutional layers of the generator. Furthermore, deep structures and batch normalization tended to result in monotonous text synthesis. Therefore, we designed the generator's layers to be wide rather than deep without batch normalization.

We also observed that the convergence of TESGAN depends on the parameter update rate of the discriminators and the generator. As in Algorithm 1, the discriminators update their parameters only during odd training epochs to allow the generator to catch up with the discriminator's learning because the convergence of the generator is commonly slower than that of the discriminator. When the discriminators updated their parameters at every training epoch, the same as the generator, adversarial training became unbalanced. Additionally, we conducted further experiments by changing the update frequency of the generator from once to three times per mini-batch step. When the generator updated only once per step, the same as the discriminators, it could not keep up with the learning of the discriminators. On the other hand, when the generator updated three times per step, the discriminators could not keep up with the learning of the generator. Therefore, we chose to update the generator twice per step, resulting in the generator being updated four times more frequently per two epochs than the discriminators, as explained in Algorithm 1.

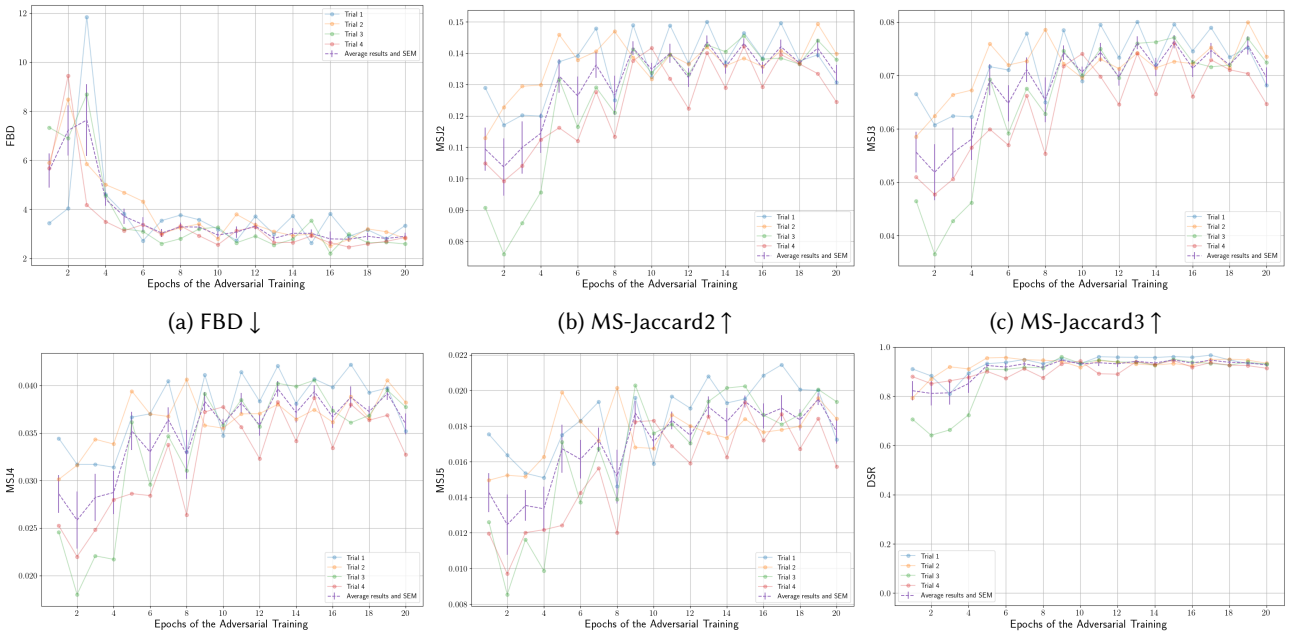
6.2 Ablation Study

In this section, we confirm the effect of the four objective functions in Section 3.4, and the results are shown in Table 4. When Seed Order Discriminator (SOD) and Seed Distribution Prediction (SDP) were not used, there was a significant difference in the results, indicating that SOD and SDP are important for high-quality text synthesis. Since MSJ evaluates text based on the n-gram of tokens, the order of the synthesized text is important. Accordingly, the MSJ results of the "w/o SOD"



(a) Zero-shot results (b) Non-zero-shot results (c) DSR, LM, SBL results

Figure 7: Zero-shot, non-zero-shot results of IMDb-trained TESGAN. DSR, LM, and SBL results of IMDb-trained TESGAN are normalized for ease of viewing.



(a) FBD ↓ (b) MS-Jaccard2 ↑ (c) MS-Jaccard3 ↑

(d) MS-Jaccard4 ↑ (e) MS-Jaccard5 ↑ (f) DSR ↑

Figure 8: Illustration of TESGAN training results. TESGANs show similar trends for every trial, and SEMs decrease during adversarial training.

in Table 4 are worse than those of the "w/o Seed Structure Discriminator (SSD)", which proves that SOD can capture the token order representations. The four objective functions were used to achieve a good overall result, demonstrating that each of the four objective functions is playing a unique role.

6.3 Activation Function Study

The results varied depending on the activation functions used at the end of the seed-making process. We conducted experiments on sigmoid, tanh, and non-use cases during the seed-making process, and their results are shown in Table 5. Table 6 shows the quality of the synthetic text for tanh and non-use cases, and the results are worse than those using sigmoid in Table 2. However, according to Table 5, the DSR results of the non-use case are higher than the sigmoid case. Thus, we can see that a higher DSR does not always mean good quality because DSR only considers data memorization. Therefore, we select the model using the sigmoid activation function, which has better results for FBD and MSJ, and moderately high DSR.

6.4 Data Memorization Study

The pre-trained GPT-2, which has 124M parameters and is used as the seed interpretation model, has been trained on relatively large corpora. Therefore, we need to confirm whether the low data memorization comes from transfer learning or the TESGAN framework. We trained three smaller seed interpretation models from

Method	FBD ↓	MSJ2 ↑	MSJ3 ↑	MSJ4 ↑	MSJ5 ↑	DSR	LM∗ ↓
TESGAN w/o SSD (13)	**2.8346**	0.1377	0.0744	0.0394	0.0204	0.9497	**4.1833**
TESGAN w/o SOD (2)	4.7724	0.1125	0.0596	0.03115	0.0164	0.8656	4.7011
TESGAN w/o SDP (13)	38.1301	0.0580	0.0252	0.0099	0.0041	0.7390	-
TESGAN w/o SFP (16)	2.9202	0.1477	**0.0790**	**0.0422**	0.0209	0.9463	4.2339
TESGAN (17)	2.8994	**0.1496**	0.0789	**0.0422**	**0.0214**	**0.9669**	4.2361

Table 4: Results of the ablation study. Numbers in parentheses indicate the training epoch of the selected model. ∗ denotes a metric not considered when selecting the best model. - denotes that the confidence of the result is low because the quality of the synthesized sentence is poor.

Activation		FBD ↓	MSJ2 ↑	MSJ3 ↑	MSJ4 ↑	MSJ5 ↑	DSR ↑	LM∗ ↓
TESGAN	None	47.261	0.108	0.060	0.032	0.016	**0.982**	-
	Tanh	9.780	0.110	0.057	0.030	0.015	0.871	5.402
	Sigmoid	**2.899**	**0.150**	**0.079**	**0.042**	**0.021**	0.967	**4.236**
P-TESGAN	None	54.002	0.111	0.059	0.030	**0.015**	**0.958**	-
	Tanh	20.158	0.118	0.061	0.031	**0.015**	0.937	-
	Sigmoid	**2.274**	**0.131**	**0.066**	**0.032**	0.014	0.841	**3.642**

Table 5: Performance according to the activation functions of the generator. ∗ denotes a metric not considered when selecting the best model. - denotes that the confidence of the result is low because the quality of the synthesized sentence is poor.

TESGAN with Tanh	**P-TESGAN with Tanh**
You are a little	You ’ re a book?
I ’ m sorry to see you off. You ’ Ve come	You are late.
I ’ m sorry.	I ’ m doing ’ t “ all day ’ s
You ’ d like a tour to see the dentist.	I don’t know what time it is?
You are late.	I ’ m sorry to hear this!
TESGAN without activation	**P-TESGAN without activation**
I ’ d like to say it! I ’ d like to	I ’ s a big, that ’ s right.
Yes, do you want to buy?	I like the back ones. They look like a shop.
I ’ s right over there?	I ’, this ’, this ’! be real,
What’s the matter?	I have a problem with my English textbooks.
I got a bite the food?	I ’ s faster, George. I ’ d like to go

Table 6: Synthesized sentences by tanh and non-use cases in Table 5. P-TESGAN denotes the perturbed TESGAN.

scratch to measure the data memorization and they have 54M, 75M, and 96M parameters each. As shown in Figure 9, DSR is high regardless of the number of model parameters during adversarial training, indicating that the low data memorization comes from the TESGAN framework.

7 Conclusion

In this work, we proposed a novel unsupervised text synthesis framework, TESGAN. TESGAN facilitated the gradient backpropagation of natural language discrete tokens by creating a continuous text embedding space called a seed. In most text-GAN studies, data memorization had been inevitable because the generator had to be pre-trained with an autoregressive approach before adversarial training. Therefore, we introduced TESGAN, which mitigated the data memorization issue by applying an unsupervised GAN framework that does not directly refer to the training data. TESGAN improved text synthesis performance during adversarial training and resulted in the best or comparable results in terms of evaluation metrics. Additionally, TESGAN exhibited the lowest data memorization ratio, and the data memorization study confirmed that these results were attributable to the TESGAN framework. Furthermore, TESGAN achieved the highest scores in human evaluations. The ablation study highlighted the importance of the four objective functions, and the synthetic text results from a large dataset-trained TESGAN demonstrated its general applicability. This paper underscores the potential of continuous embedding spaces in conjunction with discrete tokens for text

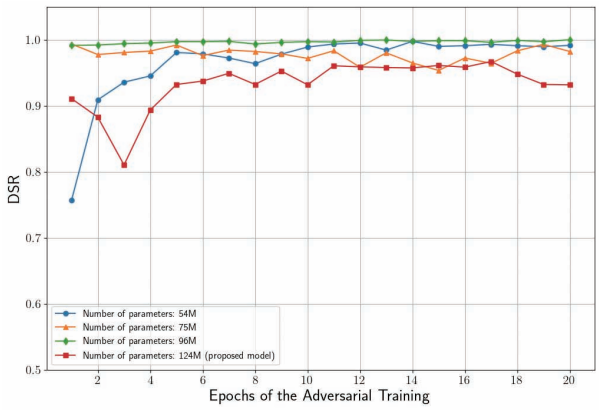


Figure 9: DSR results according to scales of seed interpretation model.

synthesis through unsupervised learning. By integrating the concept of viewing text as a continuous space with publicly available Large Language Models (Touvron et al., 2023), models can synthesize more expressive sentences, and we anticipate that many follow-up studies will emerge.

References

Alayrac, Jean-Baptiste, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. 2022. Flamingo: a visual language model for few-shot learning.

Alihosseini, Danial, Ehsan Montahaei, and Mahdieh Soleymani Baghshah. 2019. Jointly measuring diversity and quality in text generation models. In *Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation*, pages 90–98, Minneapolis, Minnesota. Association for Computational Linguistics.

Antoniou, Antreas, Amos Storkey, and Harrison Edwards. 2018. Data augmentation generative adversarial networks.

Arjovsky, Martin, Soumith Chintala, and Léon Bottou. 2017. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223. PMLR.

Bowles, Christopher, Liang Chen, Ricardo Guerrero, Paul Bentley, Roger Gunn, Alexander Hammers, David Alexander Dickie, Maria Valdés Hernández, Joanna Wardlaw, and Daniel Rueckert. 2018. Gan augmentation: Augmenting training data using generative adversarial networks.

Bowman, Samuel R., Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21, Berlin, Germany. Association for Computational Linguistics.

Caccia, Massimo, Lucas Caccia, William Fedus, Hugo Larochelle, Joelle Pineau, and Laurent Charlin. 2020. Language gans falling short. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Che, Tong, Yanran Li, Ruixiang Zhang, R Devon Hjelm, Wenjie Li, Yangqiu Song, and Yoshua Bengio. 2017. Maximum-likelihood augmented discrete generative adversarial networks.

Chen, Liqun, Shuyang Dai, Chenyang Tao, Dinghan Shen, Zhe Gan, Haichao Zhang, Yizhe Zhang, Ruiyi Zhang, Guoyin Wang, and Lawrence Carin. 2018. Adversarial text generation via feature-mover's distance. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 4671–4682, Red Hook, NY, USA. Curran Associates Inc.

Chung, Hyung Won, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models.

Dai, Wenliang, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale Fung, and Steven Hoi. 2023. Instructblip: Towards general-purpose vision-language models with instruction tuning.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Diao, Shizhe, Xinwei Shen, Kashun Shum, Yan Song, and Tong Zhang. 2021. TILGAN: Transformer-based implicit latent GAN for diverse and coherent text generation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4844–4858, Online. Association for Computational Linguistics.

Fedus, William, Ian J. Goodfellow, and Andrew M. Dai. 2018. Maskgan: Better text generation via filling in the _____. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Graves, Alex. 2013. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850.

Guo, Jiaxian, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. 2018. Long text generation via adversarial training with leaked information. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Heusel, Martin, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Hinton, Geoffrey, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*.

Hochreiter, Sepp and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Jang, Eric, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Karras, Tero, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. 2021. Alias-free generative adversarial networks.

Karras, Tero, Samuli Laine, and Timo Aila. 2018. A style-based generator architecture for generative adversarial networks.

Kingma, Diederik P. and Jimmy Ba. 2014. Adam: A method for stochastic optimization. Cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

Kingma, Diederik P. and Max Welling. 2014. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Kullback, S. and R. A. Leibler. 1951. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79 – 86.

Lester, Brian, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Li, Junnan, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models.

Li, Yanran, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017. DailyDialog: A manually labelled multi-turn dialogue dataset. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 986–995, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Lin, Chin-Yew. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Lin, Kevin, Dianqi Li, Xiaodong He, Zhengyou Zhang, and Ming-Ting Sun. 2017. Adversarial ranking for language generation. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 3158–3168, Red Hook, NY, USA. Curran Associates Inc.

Lu, Lu, Yeonjong Shin, Yanhui Su, and George Em Karniadakis. 2019. Dying relu and initialization: Theory and numerical examples. *ArXiv*, abs/1903.06733.

Maas, Andrew L., Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Maas, Andrew L., Awni Y. Hannun, and Andrew Y. Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*.

de Masson d'Autume, Cyprien, Mihaela Rosca, Jack W. Rae, and Shakir Mohamed. 2019. Training language gans from scratch. In *Neural Information Processing Systems*.

Nie, Weili, Nina Narodytska, and Ankit Patel. 2019. Relgan: Relational generative adversarial networks for text generation. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Ouyang, Long, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training

language models to follow instructions with human feedback.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Press, Ofir, Amir Bar, Ben Bogin, Jonathan Berant, and Lior Wolf. 2017. Language generation with recurrent generative adversarial networks without pre-training. *CoRR*, abs/1706.01399.

Radford, Alec, Luke Metz, and Soumith Chintala. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks.

Radford, Alec, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2018. Language models are unsupervised multitask learners.

Sandfort, Veit, Ke Yan, Perry Pickhardt, and Ronald Summers. 2019. Data augmentation using generative adversarial networks (cyclegan) to improve generalizability in ct segmentation tasks. *Scientific Reports*, 9.

Santoro, Adam, Ryan Faulkner, David Raposo, Jack Rae, Mike Chrzanowski, Théophane Weber, Daan Wierstra, Oriol Vinyals, Razvan Pascanu, and Timothy Lillicrap. 2018. Relational recurrent neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 7310–7321, Red Hook, NY, USA. Curran Associates Inc.

Sennrich, Rico, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Touvron, Hugo, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models.

Tran, Ngoc-Trung, Viet-Hung Tran, Ngoc-Bao Nguyen, Trung-Kien Nguyen, and Ngai-Man Cheung. 2021. On data augmentation for gan training. *Trans. Img. Proc.*, 30:1882–1897.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Wang, Xin, Yi Qin, Yi Wang, Sheng Xiang, and Haizhou Chen. 2019. Reltanh: An activation function with vanishing gradient resistance for sae-based dnns and its application to rotating machinery fault diagnosis. *Neurocomputing*, 363:88–98.

Wei, Jason, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2021. Finetuned language models are zero-shot learners. *CoRR*, abs/2109.01652.

Wen, Tsung-Hsien, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1711–1721, Lisbon, Portugal. Association for Computational Linguistics.

Williams, Ronald J. and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280.

Yu, Lantao, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1).

Zhang, Yizhe, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. 2017. Adversarial feature matching for text generation. In *International Conference on Machine Learning*, pages 4006–4015. PMLR.

Zhu, Yaoming, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A benchmarking platform for text generation models. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, page 1097–1100, New York, NY, USA. Association for Computing Machinery.

A Details of Models

A.1 Seed Interpretation Model

The seed interpretation model $f_\theta(\cdot)$ is necessary to predict subsequent sentences a seed makes. Therefore, the seed interpretation model must be trained with multi-turn sentences in an autoregressive way. Our interpretation model inherits the 12-layer GPT-2, derived from the decoder of the transformer language model (Vaswani et al., 2017), and has 124M parameters. We used the model achieving the highest NLTK BLEU-4[6] in the validation set of each dataset in Table 7 as the seed interpretation model.

A.2 Generator

The generator $g_\phi(\cdot)$ consists of two 1D transposed convolutional layers and has 3.3M parameters. The first and the second layers conduct convolution with 128 and 16 filters, respectively. Since sentences vary according to types of tokens and their order, forms of the real seed H_{real} are also varied. The generator generates seeds from the uniform distribution noise X with an interval of $[-10, 10)$ to make diverse forms of the seeds. Furthermore, the fake seeds generated by the deep convolutional layers and batch normalization results tend to synthesize only monotonous sentences. Thus, layers of the generator are constructed not deeply but widely and the generator does not have batch normalization layers. Leaky ReLU is used as the activation function between the two convolutional layers.

A.3 Seed Structure Discriminator (SSD)

Sentence structure is important for constructing a complete sentence. For example, "*I love you so much*" is structurally error-free, but "*I love like so much*" and "*I love*" are not. Because real seeds are created from perfect sentences, they retain the structural representation of sentences. Therefore it is important that the fake seeds should capture the structural features of the real seeds. We assume that every sentence can be the first sentence in multi-turn cases. Thus, the real seeds are obtained from sentences where the $[CLS]$ token is inserted at the beginning like Equation 3. We use the 2-layer BERT $d_\alpha(\cdot)$ to capture the structural features of sentences, and the $[CLS]$ token's feature is used to predict whether the seed is real (label 1) or fake (label 0). In addition, real and fake seeds do not pass through the embedding part of the BERT because they are already embedding spaces. Finally, the BERT used in SSD has 54M parameters.

[6] https://www.nltk.org/_modules/nltk/translate/bleu_score.html

A.4 Seed Order Discriminator (SOD)

The order of tokens is important for constructing sentences. For example, "*I love you so much*" is syntactically correct, but "*I you love so much*" and "*I love you much so*" are not. We use a 2-layer Bidirectional LSTM to consider both forward and backward directions of sentences and the model has 24M parameters. The concatenated hidden states of the last token ($[SEP]$ or $[PAD]$) and the first token ($[CLS]$) are used to predict whether the seed is real (label 1) or fake (label 0).

B Loss Function

Here, we show whole loss functions of TESGAN:

Seed Interpretation Model :

$$\mathcal{L}_{LM} = -\frac{1}{N}\sum_{n=1}^{N}\log\frac{exp(x_{n,y_n})}{\sum_{c=1}^{C}exp(x_{n,c})}$$

Adversarial Training Training

d − step :

$$\mathcal{L}_D = -\frac{1}{N}\sum_{i=1}^{N}\Big\{\big[y_i\log x_i + (1-y_i)\log(1-x_i)\big]_{SSD}^{real/fake} + \big[y_i\log x_i + (1-y_i)\log(1-x_i)\big]_{SOD}^{real/fake}\Big\}$$

g − step :

$$\mathcal{L}_G = -\frac{1}{N}\sum_{i=1}^{N}\Big\{\big[y_i\log x_i + (1-y_i)\log(1-x_i)\big]_{SSD}^{fake} + \big[y_i\log x_i + (1-y_i)\log(1-x_i)\big]_{SOD}^{fake} + \sigma(f_\theta(H_{real}))\log\frac{\sigma(f_\theta(H_{real}))}{\sigma(f_\theta(H_{fake}))} + \|\mu_r - \mu_f\|_2^2 + \|H_{real} - H_{fake}\|_1\Big\} \tag{15}$$

The first loss function in Equation 15 is cross-entropy and is used to train the seed interpretation model. The loss functions used in adversarial training operate differently in the discriminator and generator steps. In the discriminator step (d-step), the loss function is designed to train the discriminator to distinguish between real and fake seeds, predicting them as 1 and 0, respectively. On the other hand, in the generator step (g-step), the loss function aims to train the generator to predict fake seeds as 1. Additionally, SDP and SFP losses are added to assist the generator learning during the g-step.

C Statistics of Datasets

Table 7 shows the statistics of the two datasets used in this paper. We excluded single-turn reviews when constructing the IMDb multi-turn dataset. For the baseline performance experiments, we generated fake seeds equal to the number of sentences in the DailyDialog dataset to conduct TESGAN training (adversarial

training). Similarly, for the general applicability experiments, we generated nearly 300k fake seeds to conduct the experiments with IMDb datasets.

D Hyperparameters

The TESGAN framework has two training steps. The first step is seed interpretation model training. The multi-turn data for seed interpretation model training were limited to a maximum of four and eight turns in performance (DailyDialog-trained) and general applicability (IMDb-trained) experiments, respectively. Also, the maximum length of the sentence was set to 16 and 32 for each experiment (total sequence length of each experiment was 64 and 256). The 12-layer GPT-2 is used as the seed interpretation model, and both hidden and embedding dimensions are 768. We adopted the byte-pair-encodings (BPE) (Sennrich et al., 2016) tokenizer with 50,260 vocabularies in the seed interpretation model. We used the Adam optimizer (Kingma and Ba, 2014) with $1e^{-3}$ learning rate to train the seed interpretation model and set the mini-batch size to 100.

In the adversarial training phase, the sentences used as the seeds are composed of tokens explained in Equation 3, and the length of each sentence is set to 16 including special tokens. The discriminators are updated by the loss of SSD and SOD during adversarial training. Also, the generator is updated not only by the loss of SSD and SOD but also that of SDP and SFP.

The fake seeds are generated from the uniform distribution noise X with an interval of $[-10, 10)$ by the generator, which has two convolutional processes. In addition, the Leaky ReLU with slope 0.5 and the sigmoid are used in the middle and the end of the generator, respectively. We used the Adam optimizer with $2e^{-4}$ learning rate when training DailyDialog because the generator has difficulty converging when the learning rate exceeds $4e^{-4}$. However, when training on IMDb, a larger dataset than DailyDialog, we set the learning rate to $5e^{-4}$. The BERT and the LSTM models, used as SSD and SOD respectively, consist of two layers and 768 hidden dimensions. Both discriminators used the Adam optimizer with $5e^{-4}$ and $1e^{-3}$ learning rate, respectively. When the learning rates of the discriminators were larger than the proposed values, adversarial learning was imbalanced. Also, the mini-batch size was set to 128 during adversarial training. Lastly, all the above experiments took place on a machine with Ubuntu 18.04.5 and an NVIDIA RTX 3090 GPU.

E Additional Results

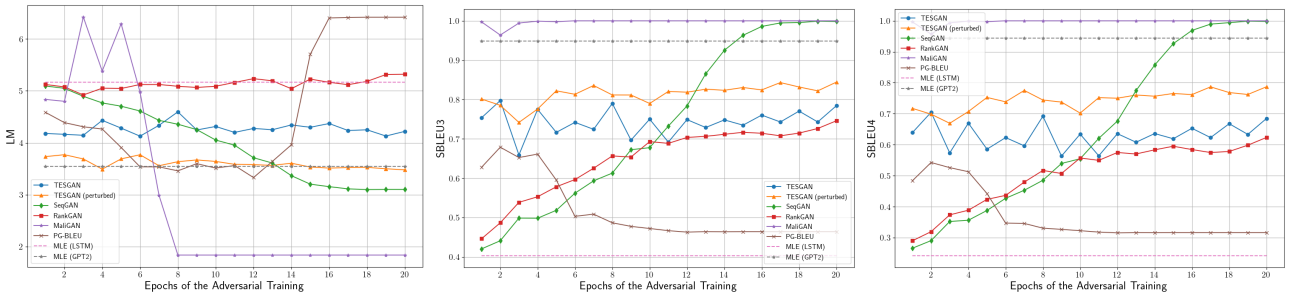
We provide evaluation results of the text generated by each model per epoch. Table 8 shows the results of 1, 5, 10, 15, 17, and 20-epoch results of each model. Also, Figure 10 shows LM and SBL results of the TESGAN-based models and the baselines. In Figure 10, the SBL results of the baselines tend to increase.

Statistics	DailyDialog			IMDb		
	Train	Validation	Test	Train	Validation	Test
# of multi-turn set	11,118	1,000	1,000	24,890	12,500	12,390
Total sentences	87,170	8,069	7,740	299,137	150,369	148,768
Avg. # of turns per set	7.84	8.07	7.74	12.02	12.03	12.01
Avg. # of words per sentence	11.30	11.21	11.44	19.34	19.40	19.28
Avg. # of tokens per sentence	14.51	14.39	14.69	24.25	24.31	24.20

Table 7: Statistics of datasets

Method	Epoch	FBD ↓	MSJ2 ↑	MSJ3 ↑	MSJ4 ↑	MSJ5 ↑	DSR (R_{syn}, R_{unq}) ↑
TESGAN	1	3.441	0.129	0.067	0.034	0.018	0.911 (1, 0.836)
	5	3.826	0.137	0.072	0.037	0.018	0.932 (1, 0.873)
	10	3.185	0.132	0.069	0.035	0.016	0.932 (0.999, 0.873)
	15	**2.624**	0.146	**0.080**	0.041	0.020	0.961 (1, 0.925)
	17	2.899	**0.150**	0.079	**0.042**	**0.021**	**0.967** (1, 0.936)
	20	3.339	0.131	0.068	0.035	0.017	0.932 (1, 0.872)
P-TESGAN	1	6.146	0.112	0.057	0.029	0.013	0.803 (1, 0.671)
	5	3.746	0.118	0.060	0.030	**0.016**	0.801 (0.998, 0.669)
	10	2.274	**0.131**	**0.066**	**0.032**	0.014	**0.841** (0.997, 0.727)
	15	**2.132**	0.121	0.061	0.030	0.014	0.812 (1, 0.683)
	17	2.274	0.122	0.063	0.031	0.014	0.789 (0.998, 0.653)
	20	2.309	0.119	0.058	0.029	0.014	0.784 (0.997, 0.646)
SeqGAN	1	**6.153**	**0.185**	**0.091**	**0.040**	**0.015**	**0.880** (0.883, 0.877)
	5	6.373	0.121	0.065	0.032	0.014	0.602 (0.639, 0.568)
	10	9.432	0.064	0.034	0.017	0.007	0.289 (0.34, 0.251)
	15	18.471	0.012	0.006	0.003	0	0.019 (0.025, 0.015)
	17	24.504	0.004	0.002	0.001	0	0.004 (0.006, 0.003)
	20	28.048	0	0	0	0	0.001 (0.013, 0.001)
RankGAN	1	**6.409**	**0.189**	**0.096**	**0.048**	**0.023**	**0.890** (0.895, 0.886)
	5	6.778	0.160	0.084	0.04	0.016	0.82 (0.851, 0.791)
	10	10.862	0.138	0.074	0.037	0.018	0.739 (0.799, 0.687)
	15	9.732	0.118	0.060	0.030	0.015	0.699 (0.791, 0.625)
	17	9.893	0.115	0.058	0.028	0.012	0.696 (0.792, 0.62)
	20	12	0.114	0.056	0.026	0.011	0.721 (0.836, 0.634)
MaliGAN	1	21.436	**0.015**	**0.006**	**0.003**	0	**0.030** (1, 0.015)
	5	**16.589**	0.003	0	0	0	0.027 (1, 0.014)
	10	57.769	0	0	0	0	0 (1, 0)
	15	57.769	0	0	0	0	0 (1, 0)
	17	57.769	0	0	0	0	0 (1, 0)
	20	57.769	0	0	0	0	0 (1, 0)
PG-BLEU	1	**9.002**	**0.071**	**0.036**	**0.015**	**0.006**	**0.569** (0.555, 0.584)
	5	18.974	0.017	0.008	0	0	0.139 (0.472, 0.082)
	10	21.568	0.001	0.001	0	0	0.041 (0.792, 0.021)
	15	89.764	0	0	0	0	0.004 (0.773, 0.002)
	17	142.384	0	0	0	0	0.003 (1, 0.001)
	20	142.384	0	0	0	0	0.001 (1, 0.001)

Table 8: Performance of each model per epoch.



(a) LM ↓ (b) SBL3 ↓ (c) SBL4 ↓

Figure 10: LM, SBL results of TESGAN-based models and baselines trained with DailyDialog.