Research on Generative Adversarial Networks for Unconditional Text Generation

Jiao Ziyun
焦 子韵

A Thesis submitted to Tokushima University in partial fulfillment of the requirements for the degree of doctor of Philosophy

September 2022

Department of Information Science and Intelligent Systems
Graduate School of Advanced Technology and Science
Tokushima University, Japan

Contents

List of Figures

List of Tables

Acknowledgment

While writing this thesis, I got the help of many tutors and classmates, and I was deeply moved. Before I finish the paper, I would like to take this opportunity to express my sincerest gratitude.

Firstly, I would like to express my sincere gratitude to my supervisor Prof. Fuji Ren for the continuous support of my Ph.D. study and related research. His serious scientific attitude, rigorous academic spirit, noble morality, and unpretentious and approachable personality left an unforgettable imprint on me. He provided me with an excellent research environment. With his patient guidance and encouragement, I have had the chance to attend international conferences and publish papers in authoritative journals. These research experiences broadened my horizon and benefited me throughout my life.

I would thank Prof. Kita, Prof. Fuketa, and Prof. Shishibori for their time and effort in reviewing my thesis. Their valuable suggestions helped improve this thesis. I would like to thank all teachers and classmates in Ren Lab for their help during this period.

I want to thank Prof. Xin Kang, who gave me strong guidance in my dissertation topic selection and preparation stage and excellent support and encouragement for my chosen research field. At the same time, he made suggestions, encouraging me to work hard to complete the thesis with a positive attitude.

Meanwhile, I would like to thank my classmates in the A1 group. Thanks to Mr.Yangyang Zhou, and Mr. Zheng Liu, for their valuable suggestions and shared experience in research.

Finally, I would like to thank my family for their silent support and encouragement to me. I would like to thank you for your support and care in my study journey. Your encouragement is a powerful driving force for me to keep moving forward.

Abstract

Language and writing play an irreplaceable role in human communication as natural products of civilization. As a branch of natural language processing (NLP), natural language generation (NLG) has received extensive attention since its inception. In the process of human communication, NLG and natural language understanding (NLU) are the two most essential components. In modern human-computer interactions, NLG is also a core functional requirement of machines. As an automated process that generates human-readable text from input information with specific interaction goals, NLG employs different inputs for different tasks. From the perspective of input information, NLG can be classified as text-to-text, data-to-text, multimodality-to-text, or zero-to-text, also known as unconditional text generation. Because no input is provided in the task of unconditional text generation, the model is required to generate natural language text freely. The Generative Adversarial Network (GAN) for text is a standard model for unconditional text generation tasks.

Initially proposed in 2014, GANs have been widely used in Computer Vision (CV) tasks. However, the development of GANs for text generation has progressed slowly. On one hand, the guidance information passed by a discriminator to the generator is generally extremely weak. On the other hand, gradients cannot be transferred appropriately between the generator and discriminator, which prohibits normal gradient-based training. In response to these issues, the key contributions of this thesis are summarized below.

(1) Compared with the conventional loss function, the Wasserstein Distance can provide more information to the generator. We proposed a new architecture based on RelGAN and WGAN-GP, dubbed WRGAN. The discriminator network structure of WRGAN uses the 1-dimensional convolution of multiple kernel sizes and residual modules. Correspondingly, we adjusted the network's loss function with the gradient penalty Wasserstein loss. This thesis provides and analyzes the experimental results on

multiple datasets and the influence of hyperparameters on the model. The experiments demonstrated that our model outperformed most current models on real-world data.

(2) We improved TILGAN for unconditional text generation by refactoring the generator. In short, we implemented Multi-head Self-Attention to replace the linear and BN layers to endow the generator with superior text generation capabilities. Our model consists of three components: a Transformer autoencoder, a Multi-head Self-Attention-based generator, and a linear discriminator. In the transformer autoencoder, the encoder component encodes the distribution of real samples, whereas the decoder decodes real or generated sentence vectors into text. The loss functions for autoencoder and GAN are cross entropy and KL divergence, respectively. On the MSCOCO and EMNLP WMT News datasets, the proposed model has achieved a higher BLEU score than TILGAN. Our ablation experiments also demonstrate the effectiveness of the proposed generator network for the unconditional text generation.

Chapter 1

Introduction

1.1 Motivation

Natural Language Generation (NLG) is essential research in the field of Natural Language Processing (NLP) [1]. NLG system can be defined as generating readable textual representations using information that is not limited to textual forms as input. NLG can be regarded as the inverse of Natural Language Understanding (NLU). The NLU system needs to clarify the meaning of the input sentence and generate a machine expression language. The goal of the NLG system is how to translate concepts into readable language. The working process starts from the abstract concept level and generates text by selecting and executing specific semantic and grammatical rules [2].

Achieving high-quality NLG is a vital sign of the gradual maturity of artificial intelligence technology. NLG technology has a wide range of applications, such as intelligent question answering, human-machine dialogue, machine translation, automatic generation of advertising words, and automatic news generation [3]. NLG reduces the difficulty of human-machine dialogue. It has a huge impact on our daily life and work, and has become a typical project in both academia and industry. Text generation systems such as automated insights, narrative science, and the “Xiao Nan” and “Xiao Ming” robots have been put into use. With the development of technology, many excellent models and extensive cutting-edge research have emerged in the field of NLG, but there are still many problems to be solved. For example, natural language

is not standardized. Although we can observe some basic rules, natural language is still too flexible. The same meaning can be expressed in many ways. This causes difficulty for NLG, whether the aim is to understand natural language based on rules or learn the inherent characteristics of data through machine learning. Thus, for NLG systems, computational methodologies need to be continuously developed and optimized for more natural and smooth text generation.

In general, NLG systems are an essential research field in NLP and still have many application prospects and research values. For this reason, we chose NLG content for our research.

1.2 Background

NLG is an essential research branch in NLP with a long history. In the 1950s, NLG was first proposed as a sub-problem of machine translation [4]. In the 1970s, NLG began to generate simple explanations for expert systems and to write natural language answers for the results returned by database queries. In the early 1980s, NLG gradually emerged as an independent research field within NLP, and researchers began to explore its unique concerns and research questions. In the 1980s and 1990s, researchers proposed statistical language models and began to describe language and characters from the perspective of probability and statistics. Since then, the era of statistical language models has already started. In 2003, Bengio [5] proposed a feedforward neural network, which changed the modeling ideas of traditional language models. In 2013, due to the proposal of the word vector [6], language modeling based on the neural network began to appear in large numbers. Today, language models based on neural networks have dominated NLG methods.

1.2.1 Language Models

NLG is broadly defined as the automated process of generating human-readable language text from given input information under a specific interaction goal. NLG has different inputs depending on the task setting, but the output must be readable natural language text.

According to different generation ideas, NLG models can be roughly divided into the traditional pipeline [7] and the end-to-end [8] models based on neural networks.

(1) The pipeline model includes multiple independent steps, and the data are processed by each module to obtain the final output. The classic pipeline model is mainly divided into six steps [9]. The first step is content determination [10]. The purpose of this step is to determine what information should be included and excluded in the generated text. For example, in a ticket booking system, the information about the ticket is obtained by querying the ticket. This information is the content that should be included in the output statement. The second step is text structure [11], which aims to organize the order of the text reasonably or reasonably arrange which information is displayed first and which information is displayed later. The third step is sentence aggregation [12]. Not every piece of information needs to be expressed in one sentence. The function of this step is to cluster the information and express the information that can be combined in one sentence. Through the first three steps, it is determined how many sentences need to be generated, what information each sentence contains, and in what order the information is expressed. The fourth step is grammaticalization [13], which introduces some connecting words to facilitate the formation of a sentence later. The fifth step is reference expression generation [14], and the purpose is to select words related to the content domain for modification and adjustment. At this moment, each sentence is still a collection of words and does not constitute a real sentence. So the final step is to formally convert the previous set of sentences into a complete well-structured sentence [15, 16]. Note that the model proposed in a 2002 paper called plan-based NLG [17] simplifies the text generation process into three stage: sentence plan generation,

sentence plan reranking, and surface realization. In the sentence plan generation stage, the sentence planning tree is generated, each node is an action of the dialogue, and the sentence planning tree is converted into the final sentence, which is to be generated in the surface realization stage. Although plan-based NLG simplifies the generation stage, it still has the disadvantages of pipeline models. The quality of the results of the previous step directly affects the next step, thus affecting the results of the entire training. In addition, this model requires a great investment of manpower and time for the manual annotation of specific fields, and it is difficult to extend to new fields.

(2) End-to-end models mainly refer to the NLP model developed based on neural networks. When end-to-end models handle the natural language standardization problem, they no longer divide the sub-problems manually, but include the intermediate operations in the neural network, eliminating the need for costly and error-prone data labeling. End-to-end models increase the overall fit of the model and improves the efficiency of the system in solving problems by reducing manual preprocessing. In recent years, the NLG field has generally used the end-to-end method. However, the end-to-end NLG generation framework lacks both the explicit utilization of linguistic knowledge and the effective means to control the quality of the generated content, which is not suitable for situations where data are lacking. The advantages and disadvantages of excellent text generation models in recent years and a brief introduction of each model are listed in Table 1.1.

Table 1.1 The excellent text generation models in recent years.

Models	Introduction	Advantages and Disadvantages
RNN [18]	The Recurrent Neural Network (RNN) passes the sequence information of each item through the feedforward network. It uses the output as the input of the next item in the sequence, and each item stores the previous step's information.	A: Get sequence features of input data. D: Cannot generate long sentences and parallel compute.
LSTM [19]	The Long Short-Term Memory (LSTM) networks and their variants can solve the vanishing gradient problem and generate longer sentences. They handle the dependencies in long sequences of inputs more accurately.	A: Solve the gradient disappearance problem and get longer sentences. D: Cannot parallel compute.
Seq2Seq [20, 21]	The Sequence-to-Sequence (Seq2Seq) is generally based on the encoder-decoder framework. It was proposed to solve the problem that most sequence lengths are not equal. For example, in machine translation, the source language sentences and the target language often do not have the same length. The model is good at using the sequence's global information, synthesizing the sequence's context, and generating another corresponding representation sequence.	A: Handling unequal length sequences. D: Exposure Bias and decoder often fails to align encoder.

VAE [22]	The Variational Autoencoder (VAE) is a popular method for unsupervised learning of complex probability distributions. The most significant feature of VAE is to imitate the learning and prediction mechanism of the automatic encoder and to encode and decode between measurable functions.	A: Can learn latent propertiesand construct new elements. D: Tend to produce more ambiguous data.
Transformer [23]	The Transformer consists of a set of encoders and decoders. The job of the encoder is to process the input of arbitrary length and generate the encoding, and the job of the decoder is to convert the encoding to words. The Transformer uses the Self-Attention mechanism to obtain the relationship between all other words and generates an encoding for each word.	A: Directly capture the relationship between all words in a sentence (global information). D: Insensitive to word position information.
ELMo [24]	The Embeddings from Language Models (ELMo) does not use a word corresponding to a fixed vector but implements a sentence or a paragraph into the model. The model infers the word vector corresponding to each word according to the context. The ELMo adopts a typical two-stage process: The first stage is to get the pre-training language model. The second stage is to extract the word embedding of each layer corresponding to the word from the pre-training network as a new feature to add to downstream tasks.	A: Understand polysemy accurately. D: Still using LSTM.

GPT [25-27]	The Generative Pre-Training(GPT) is a typical pre-training and fine-tuning two-stage model. The pre-training stage uses massive text data to acquire linguistic knowledge through unsupervised learning. In contrast, fine-tuning uses the training data of downstream tasks to obtain models for specific tasks.	A: Perform very well on multiple NLP tasks. D: Huge amount of parameters, difficult to fine-tune.
BERT [28]	The Original BERT is not suitable for NLG tasks. However, a model called MASS combined the GPT model to form an encoder-decoder framework in 2019, supplemented by sophisticated pre-training techniques. It performed well in some NLG tasks. MASS randomly masks a contiguous segment of the sentence and then generates the segment by predicting through the encoder-attention-decoder framework.	A: Unified pre-training framework (including GPT and Bert). The decoder has strong language modeling ability. D: Large amount of calculation.

1.2.2 Tasks in Natural Language Generation

According to different task requirements and different input formats, NLG can be roughly divided into text-to-text generation, data-to-text generation, multimodality-to-text generation, and unconditional text generation. The different generation tasks and their corresponding inputs are listed in Table 1.2.

Table 1.2 Tasks of text generation and corresponding inputs.

Tasks	Input
Text-to-text generation	Text sequence
Data-to-text generation	Structured data

Multimodality-to-text generation	Image,video and voice data
Unconditional text generation	Random noise

(1) Text-to-text generation can be divided according to different tasks: document summarization [29], sentence compression [30, 31], sentence fusion [32, 33], and paraphrase generation [34, 35]. Document summarization can be divided into extractive summarization [36] and generative summarization [37]. Extractive summarization is relatively simple, and usually uses different methods to evaluate document structural units (e.g., sentences and paragraphs), assign a certain weight to each structural unit, and then select the most important structural units to form a summary. The extractive summarization method is widely used, and the structural unit usually is a sentence. Generative summarization methods usually need to use NLU technology to analyze the grammar and semantics of the text, fuse the information, and use NLG technology to generate new summary sentences. Current document summarization methods are mainly based on sentence extraction [38]. The sentence in the original text is used as a unit for evaluation and extraction. The advantages of this method are that it is easy to implement and can ensure good readability of the summary sentence. This type of method mainly includes two steps: the first is to calculate or sort the importance of the sentences in the document, and the second is to select important sentences and combine them into a final summary. There are also some researchers who study generative summarization [39]. They usually represent the original document as a deep semantic form, then conduct an analysis to obtain a deep semantic representation of the abstract, and finally generate a summary text from the deep semantic representation of the abstract, such as Abstract Meaning Representation (AMR) [40]. Such methods are relatively complex, and the current methods are still in the exploratory stage. The performance of generative summarization methods is unsatisfactory. Sentence compression

and sentence fusion techniques are generally used in text summarization systems to generate summaries with more compact information. Sentence fusion technology combines two or more related sentences with overlapping content to obtain a sentence. According to different purposes, one type of sentence fusion only retains the common information in multiple sentences and then filters irrelevant details. The other type of sentence fusion only filters out multiple sentences. The paraphrase generation technology generates a new paraphrase text by rewriting the given text. Generally, the output text and input text have the same mean but different expressions.

(2) In the task of data-to-text generation, the knowledge graphs [41] are widely used in NLP since 2020, there are many NLG methods that utilize a series of Resource Description Framework (RDF) triples, AMR graphs, or a series of table cells. They can generate coherent human-readable text such as instructions or questions. Mei [42] et al. added the aligner to select important information based on the encoder-decoder model, and proposed an end-to-end model for generating text from data based on deep learning. Song [43] et al. applied a slightly modified Transformer encoder that explicitly handles surface form relationships and then added two autoencoder losses to the standard language model losses, which are specifically designed to capture and linguistically structure the graph.

(3) There are also different tasks in multimodality-to-text generation, such as image caption, story generation, and visual question answering (VQA) [44]. On the image caption task, Vinyals et al. use an encoder-decoder-like model for generation. Xu [45] et al. further added the Self-Attention mechanism. Huang et al. proposed the task of generating stories for image sequences, and then provided three levels of datasets: descriptive text for a single image, stories for a single image, and stories for a sequence of images [46]. While VQA is a task that combines Computer Vision (CV) and NLP, given a picture and a question,

its goal is to infer the correct answer to the question from the visual information of the picture. Shih [47] et al. proposed a model based on the Self-Attention to encode pictures with VGG, use the word vector to average the problem, and finally generate the answer through a two-layer network. Wu [48] et al. proposed integrating image caption models and external knowledge bases to generate answers.

(4) On the task of unconditional text generation, the sequence GAN (SeqGAN) [49] created a mode of Generative Adversarial Network (GAN) [50] in text generation. The maximum-likelihood augmented discrete GAN (MaliGAN) [51] proposed a new loss function of the generator to replace the Monte-Carlo [52] tree search (MCTS) and was found to obtain better results. The margin-ranking GAN (RankGAN) [53] changed the original discriminator from a binary classification model to a sorting model. The long-text generation via adversarial training with leaked information (LeakGAN) [54] leaked the characteristic information of the high-level discriminator to the manager module to guide the generator to generate long text. The masked-text GAN (MaskGAN) [55] used the Actor-Critic algorithm in reinforcement learning (RL) to train the generator and then uses maximum likelihood and stochastic gradient descent to train the discriminator. The diversity-promoting GAN (DP-GAN) [56] was proposed by focusing on diversified text generation. The author improved the discriminator based on SeqGAN and proposed a discriminator based on the language model. The generating sentimental texts via mixture adversarial networks (SentiGAN) [57] has multiple generators and a multi-class discriminator. Multiple generators are trained simultaneously, aiming to generate text with different emotion labels without supervision. For a detail introduction to the model of unconditional text generation, refer to Chapter 2.

1.3 Research Objective and Content

1.3.1 Improved RelGAN with Wasserstein Loss

In this study, we proposed a new architecture based on RelGAN [58] and WGAN-GP [59] named WRGAN. We rebuilt the discriminator architecture with the 1-dimensional convolution of multiple kernel sizes and residual modules [60]. Correspondingly, we modify the generator and discriminator loss functions with the gradient penalty Wasserstein loss. Then, we made the discriminator and the generator with relational memory coordinated by Gumbel-softmax relaxation to train the GAN model on discrete data. The paper provided the experimental results on multiple datasets and analyzed the influence of hyperparameters on the model. The experiments demonstrated that our model outperformed most current models on real-world data.

1.3.2 Improved Transformer-Based Implicit Latent GAN

In this study, we improved the transformer-based implicit latent GAN (TILGAN) [61] for unconditional text generation by refactoring the generator. In short, we use Multi-head Self-Attention [62] to replace the linear and BN layers to endow the generator with better text generation capabilities. Our model consists of three components: a Transformer autoencoder, a Multi-head Self-Attention-based generator and a linear discriminator. The encoder in the Transformer autoencoder encodes the distribution of real samples. The decoder decodes real or generated sentence vectors into text. The loss functions for autoencoder and GAN are cross entropy and KL divergence, respectively. On the MSCOCO [63] and EMNLP WMT News [64] datasets, the proposed model has achieved a better BLEU [65] score than TILGAN. Our ablation experiments also verified the effectiveness of the proposed generator network for unconditional text generation.

1.4 Organizations

This thesis mainly investigates the background and research status of the NLG task and then proposes some unconditional text generation approaches for addressing some existing challenges. The organizational structure of this thesis is as follows:

Chapter 1 discusses the motivation and background of the NLG task and introduces the main research contents and organizational structure of this thesis.

Chapter 2 introduces related works on the unconditional text generation task. This chapter first introduces GAN, GAN for text generation methods, and problems of GAN for text generation. Then, the research status and progress of unconditional text generation technology in recent years are reviewed.

Chapter 3 introduces the proposed model architecture based on RelGAN and WGAN-GP, describes the training process, gives the results of the model on multiple datasets, and explores the influence of hyperparameters for the model.

Chapter 4 introduces the improved model based on TILGAN, describes the training process, and gives the results of the model on multiple datasets.

Chapter 5 concludes the thesis and discusses future works.

Chapter 2

Related Works

2.1 Unconditional Text Generation Models

According to different modeling ideas, unconditional text generation models can be roughly divided into three categories: the Generative Pre-Training (GPT) models based on Transformer, the Variational Autoencoder (VAE) models, and the Language GANs.

GPT is a typical pre-training and fine-tuning two-stage model. The pre-training stage uses massive text data to acquire linguistic knowledge through unsupervised learning, while fine-tuning aims to use the training data of downstream tasks to obtain models for specific tasks. The structure of the GPT model fulfills two important points: one is to use Transformer as a feature extractor, and the other is to use a one-way language model. With the advancement of technology, the GPT model has developed to the third generation, namely GPT-3. In addition to common NLP tasks, GPT-3 achieves impressive performance on many difficult tasks. Beyond its excellent performance, GPT-3 is also huge. GPT-3 is the largest Transformer model released to date in NLP, with 175 billion parameters, 45 TB of training data, and up to $12 million in training costs.

As a generative model comparable to GAN, VAE combines the advantages of Bayesian methods and deep learning. It has an elegant mathematical foundation, a simple and easy-to-understand architecture, and satisfactory performance. The original VAE model framework is shown in Figure 2.1.

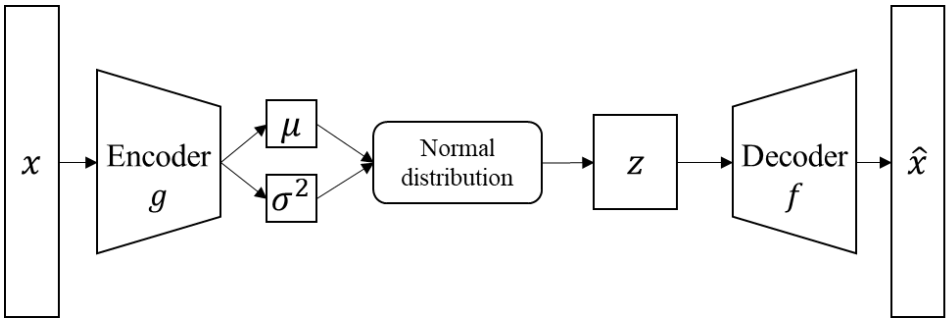

Figure 2.1 The original VAE framework.

The ability of VAE to extract disentangled latent variables also makes it more meaningful than general generative models. VAE is a kind of latent variable model. The basic idea of the autoencoder is to transform a set of real samples into an ideal data distribution through the encoder network. Then a set of generated samples is obtained by passing a decoder network. An autoencoder model is obtained when the generated samples are close enough to the real samples. VAE has achieved excellent results in multiple NLP tasks, and similar ideas to VAE can also be seen in language GANs.

Both GAN and VAE are deep generative models that can generate data with a complex distribution from random noise. However, the data are generated from different perspectives. They construct different loss function forms to measure the generated data. For VAE, the researcher believes that the data x are generated by a latent variable z, and z contains the features and information of x. For GAN, the researchers believe that the complex distribution of x can be obtained from a simple distribution $p(z)$ through a series of transformations, where z is random noise and has no physical meaning.

Traditional language models use a combination of teacher forcing and maximum likelihood in the training process. However, the traditional language models have some problems that cannot be ignored while training. Using the GAN method can effectively alleviate the problems of repetitive, short, and meaningless generation caused by the traditional language models based on the maximum likelihood estimation objective function. Therefore, many unconditional text generation models choose GAN-based methods.

2.2 GAN for Text Generation

GAN is an unsupervised learning method that was proposed by Ian Goodfellow and his colleagues in 2014. With the improvement of the theory, GAN has shown its great potential gradually. Moreover, GAN has produced many fancy CV applications, such as image generation [66], image conversion [67], style transfer [68, 69], and image restoration [70]. GAN can be leaned in an unsupervised way by letting two neural networks play against each other. GAN includes a generator and a discriminator, where the goal of the generator is to generate fake samples that can fool the discriminator. The goal of the discriminator to distinguish between the real and fake samples. In the end, the generator and the discriminator reach a Nash equilibrium in the process of playing against each other [71]. In this way, learning GAN models can essentially be thought of as a minimax game, with the objective function given by:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim P_{data}(x)}[logD(x)] + \mathbb{E}_{z\sim P_z(z)}\left[log\left(1 - D\big(G(z)\big)\right)\right], \quad (1)$$

where x represents the real sample, and z represents the random noise. The goal of the generator is:

$$arg\ max\ P\left(D\big(G(z)\big)\right), \quad (2)$$

and the goal of the discriminator is:

$$arg\ max\ P\left(D(x)\right) - P\left(D\big(G(z)\big)\right), \quad (3)$$

In the field of CV, GANs have rapidly become the hotspot in recent years owing to their superior performance [72]. However, the development has been relatively slow in NLP. There are some problems when extending the idea of GAN to text generation. The two main problems are detailed below.

(1) When GAN faces discrete data, the discriminator cannot pass the gradient to the generator through backward propagation. In Equation 2, $G(z)$ generates samples through the $arg\ max$, which is also called sampling [73]. Because this operation in text generation is a non-derivable process, gradients cannot transfer properly between the generator and the discriminator, which prohibits the normal gradient-based training [74]. For example, we assume the

vocabulary vector is $[\text{“}penguin\text{”}, \text{“}cat\text{”}]$ and the vocabulary vector corresponds to the vector $[x_0, x_1] = [1,0]$. With the $argmax(x_0, x_1)$, we always obtain the same word “$penguin$,” even if the value of x_1 increases from 0 to 0.999. In this case, the gradient of x_i is always equal to 0.

(2) The training process of GAN is unstable. We need to balance the generator and the discriminator carefully. Moreover, the generation task is much more complicated than discrimination. Simultaneously, the discriminator’s guidance to the generator is too weak [75], and the direction contains little information. For the generator, it can only obtain a “true or false” probability in return. Furthermore, the discriminator may even “cheat.” Because the real sample uses one-hot vectors, the discriminator does not need to judge whether the generating data distribution is closer to the real data [76]. It just needs to identify whether only one item of the current data is 1, and the rest iterms are all 0.

For the above problems, text GANs offer some effective solutions, such as RL for sequence generation, Gumbel-softmax relaxation [77], and Wasserstein Distance [78]. At present, GANs for text generation have been able to generate fluent texts. GANs are often used in unconditional text generation. In some tasks that need to control the generation direction, such as machine translation, dialogue generation, text summarization, GANs do not achieve excellent performance yet. Therefore, this thesis only involves unconditional text generation. The most wildly used of the evaluation datasets used for unconstrained text generation include the COCO Image Captions [79], EMNLP WMT, Chinese Poems [80], MSCOCO.

2.2.1 Reinforcement Learning for Sequence Generation

For the above problems, the first solution is the combination of GAN and RL for sequence generation [81].

For example, SeqGAN is proposed by Yu et al. with the RL algorithm. This solution deals with non-differentiable problems by considering the RL algorithm. For the discriminator’s difficulty in evaluating incomplete sequences, the author proposed to

draw on the idea of the MCTS, which can evaluate non-complete sequences at any time. The model structure is shown in Figure 2.2.

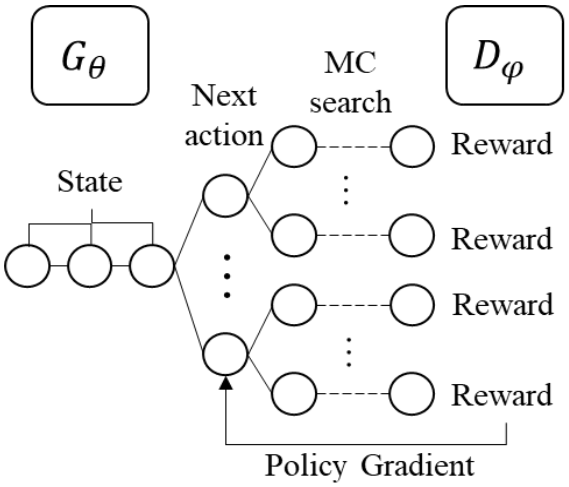

Figure 2.2 SeqGAN structure.

In Figure 2.2, G_θ represents the generator and D_φ represents the discriminator. The policy network is regarded as G_θ. The existing node is called the current state, and the next node to be generated is called action. Because D_φ needs to score a complete sequence, the MCTS completes the action sequences for obtaining the rewards. D_φ generates rewards for these complete sequences, and sends them back to G_θ.

Generally, the authors suggest using the RNN or LSTM as a generator, inputting word embedding into each node, and combining with a linear hidden layer to get the probability of outputting each word. The generator can sample a batch of generated sequences according to this probability. The MCTS selects the next action based on the probability distribution of each node. After obtaining the sampled sequences, the discriminator obtains a set of probabilities with the sampled sequences. The average value of this set of probabilities is considered the reward. When we train the generator via policy gradient, we can increase the selection probability of actions with more rewards and decrease the selection probability of actions with fewer rewards.

The LeakGAN, which is an improvement of the SeqGAN, also uses the RL algorithm for training. Unlike SeqGAN, the author additionally “leaks” some high-level information of the discriminator to a “manager.” The manager module is an LSTM

network that acts as an intermediary to help the generator to complete the generation task. Specifically, in addition to the reward given by the discriminator, the generator can additionally obtain the high-level feature representation of the discriminator at each moment. In this way, the generation of long texts is more accurate and diverse. The model structure is shown in Figure 2.3.

The starting point of LeakGAN is that the discriminator is a specific model set by humans, such as CNN, rather than a “black box” system. Because the internal structure of the discriminator is known in advance, the discriminator can provide detailed structural information to the generator. In LeakGAN, the discriminator is divided into the feature extractor and the softmax classification layer. The feature extractor obtains a high-level feature representation based on the current state and provides this representation to the “manager” as “leak” information. The manager module receives the “leak” information provided by the discriminator and obtains the action sub-goal after transformation. After that, the action sub-goal performs a dot product with the action embedding generated by LSTM. Finally, the generator uses the result to draw a sample of the next word.

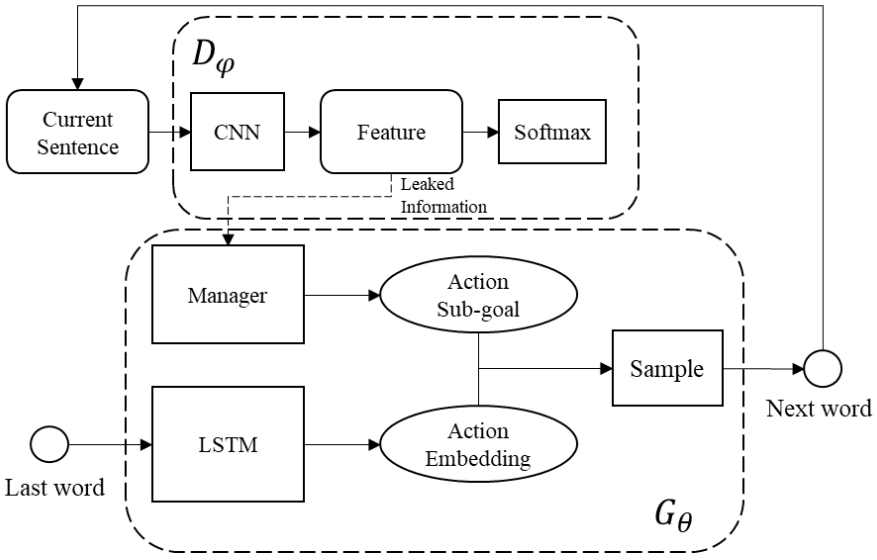

Figure 2.3 LeakGAN structure.

In the training process of SeqGAN, updating the parameters of the generator relies on the reward of the discriminator combined with RL. In addition to the scalar reward, the high-level feature representation of the discriminator can be obtained at each

moment as an additional "feature." This feature helps the generator to generate text sequences, which provides a guidance for long-term text generation.

To generate high-quality language descriptions, RankGAN was proposed. The authors relax the training of the discriminator to a learning-to-rank optimization problem. Furthermore, they turn the discriminator into a ranker. Correspondingly, the input of the ranker consists of a generated sentence and multiple human-written sentences. The goal of the ranker is to rank the generated sentence lower than human-written sentences. Therefore, the reward of RankGAN contains the ranking score. The overall optimization goal is:

$$\min_{\theta} \max_{\emptyset} \mathcal{L}(G_\theta, R_\emptyset) = \mathbb{E}_{s\sim P_h}[logR(s|U, C^-)] + \mathbb{E}_{s\sim G_\theta}\big[log\big(1 - R(s|U, C^+)\big)\big], \quad (4)$$

where U is the set of reference data for obtaining relative ordering. C^- is sampled from generated data when the input s is real data. While C^+ is sampled from real data when s is generated data. The model calculates the cosine similarity between the input s and the reference data $u\ (u \epsilon U)$ to obtain the ranking score. The definition of reward in RankGAN is the same as in SeqGAN. However, the quality of text generated by RankGAN is superior to SeqGAN due to the additional ranking information of RankGAN.

SentiGAN has multiple generators and a multi-class discriminator to address the above problems. In the SentiGAN framework, multiple generators are trained simultaneously, aiming to generate texts with different sentiment labels in an unsupervised manner. A penalty-based objective is included in the generator to force each of the generators to produce diverse examples of a specific sentiment label. Furthermore, using multiple generators and a multi-class discriminator allows each generator to focus on accurately generating its examples of specific sentiment labels. Unlike the previous work, SentiGAN uses a penalty-based loss function instead of a reward-based loss function. This strategy enables the generator to generate more diverse sentences. Given K categories of sentiments, SentiGAN uses K generators and one discriminator. The goal of the $i-th$ generator is to generate text with the $i-th$ sentiment type, fooling the discriminator as much as possible. The goal of the

discriminator is to distinguish the generated text from the real text with K categories of sentiments.

In MaskGAN, both the generator and the discriminator use the same Seq2Seq structure. The generator is trained by the RL algorithm, and the discriminator is trained by the previous maximum likelihood method. In order to solve the problem that GAN is non-differentiable in dealing with discrete data, the authors adopt a convenient Actor-Critic algorithm [82], which can fill in the missing parts of masked sentences according to the context. To address the problem of training instability in GANs, the authors treat text generation as a fill-in-the-blank or in-filling task. The text generation process can rely on the context information around the missing parts for word prediction and word filling. The generator aims to make the generated text and the corresponding real text indistinguishable from the discriminator.

Combined with the RL algorithm, the gradient problem caused by the text discretization output is alleviated. As a result, many RL-based GANs have emerged for text generation. However, a single scalar reward signal obtained from the RL algorithm leads to two problems of information sparseness and information incompleteness in training the generator. Although many methods consider making full use of the information from the discriminator and providing a highly diverse reward to the generator, the problem of information sparseness still exists. Thus, the improved methods can only alleviate the problem rather than solve it.

2.2.2 Wasserstein Distance

Firstly, we explain the distance measurement method called "Earth Mover's (Wasserstein) Distance" $W(q,p)$, which is informally defined as the minimum cost of transporting mass in order to transform the distribution q into the distribution p, where the cost is the mass times the transport distance. Under mild assumptions, $W(q,p)$ is continuous everywhere and differentiable almost everywhere. The Wasserstein Distance $W(q,p)$ can be defined as:

$$W(q,p) = \min_{\gamma} E_{(q,p)\sim\gamma}\big||q-p|\big|, \tag{5}$$

where γ is a joint probability distribution that satisfies the constraints:

$$\int \gamma(q,p)\,dq = P_g(q), \int \gamma(q,p)\,dp = P_f(p). \quad (6)$$

Compared with KL and JS divergence, Wasserstein Distance can still reflect their distance even if the two distributions do not overlap. On the one hand, the Wasserstein Distance is smooth. If we use the gradient descent method to optimize the distance parameters, KL and JS cannot provide gradients in some cases, while the Wasserstein Distance can solve this problem normally. Similarly, if the two distributions do not overlap or the overlap is negligible in a high-dimensional space, then KL and JS can neither reflect the distance nor provide gradients, while the Wasserstein Distance can provide meaningful gradients. Thus, we calculate the distance between the generated data distribution and real data distribution through the Wasserstein Distance.

To address the unstable training process of GANs, Arjovsky et al. proposed the WGAN and the WGAN-GP. They provide the theoretical solution. WGAN converts the discriminator's task from a binary classification into calculating the Wasserstein Distance. Forcing the discriminator to calculate the distance between the generated and the true data distribution prevents the discriminator from "cheating." The discriminator can also provide more accurate guidance information to the generator, not just the probability of "true or false." Because the weight clipping strategy in WGAN causes most of the weights to approach two extremes, WGAN-GP was proposed, which uses gradient penalty to replace weight clipping. This strategy makes the training more stable and produces higher-quality images.

To address the difficulty in passing the gradient, Wasserstein Distance can directly calculate the distance between the real and the generated sample distribution without the $arg\,max$. The non-derivable problem no longer exists, theoretically. WGAN completely solves the problem of unstable GAN training and no longer needs to carefully balance the training degree of the generator and the discriminator.

Unfortunately, WGAN is used more in CV. The proposal of WGAN is not aimed at solving the problems of GANs in text generation, and there are few applications in text generation. For example, when using Wasserstein loss in RelGAN, we found that the gradient would disappear, and the discriminator loss was almost equal to 0 during

training. The detailed description and the performance of RelGAN using Wasserstein loss can be found in section 3.6.6.

2.2.3 Gumbel-softmax Trick

There is a reparameterization trick in VAE that uses random sampling while ensuring that the gradient can be passed back so that the model can be trained and updated. Gumbel-softmax is also a reparameterization technique. Gumbel-softmax makes the process of sampling discrete variables derivable in an approximation.

Gumbel-softmax was first proposed for the reparameterization of categories. The improvement goal applied to GAN can be considered to design a more "powerful" softmax, which can replace the sampling operation in the original GAN.

For the discrete variables z and the distribution vector $\pi = [\pi_1; \pi_2; \dots; \pi_k]$, we can obtain the discrete variables z as:

$$z = onehot\left(\underset{i}{arg\,max}\,[g_i + log\pi_i]\right), \tag{7}$$

where g_i is a sample taken from a $Gumbel\,(0,1)$ distribution. The Probability density function (PDF) function of $Gumbel\,(0,1)$ is defined as:

$$p_{\pi,\tau}(y_1, \dots, y_k) = \Gamma(k)\tau^{k-1}\left(\sum_{i=1}^{k} \pi_i / y_i^{\tau}\right)^{-k} \prod_{i=1}^{k} (\pi_i / y_i^{\tau+1}). \tag{8}$$

The $Gumbel\,(0,1)$ distribution can be sampled using inverse transform sampling by drawing $u \sim Uniform\,(0,1)$ and computing $g = -log(-log(u))$. In order to enable the derivation of the sample for the statistical parameter, it is also necessary to solve the problem that $arg\,max$ cannot be derivable. Then the softmax is used instead of $onehot\,(arg\,max)$, the collected samples can be written as:

$$y_i = \frac{exp((log(\pi_i) + g_i)/\tau)}{\sum_{j=1}^{k} exp((log(\pi_j) + g_j)/\tau)} \quad for \quad i = 1, \dots, k, \tag{9}$$

where τ is the temperature parameter. The smaller τ is, the closer the sampling is to the result of $arg\,max$, and the closer the sample is to the one-hot vector, but the

variance of its corresponding gradient estimator is also larger. During normal training, the τ will be gradually reduced.

The typical representative network is RelGAN. To address the difficulty in passing the gradient, RelGAN utilizes Gumbel-softmax relaxation to simplify the model. RelGAN uses relational memory on the generator, which gives it more vital expression ability and better generation ability on long text. RelGAN also uses multi-layer word vector representation on the discriminator to make the generated text more diverse. Experiments showed that RelGAN achieved good results in the quality and diversity of the generated text. In addition, multi-channel feature extraction is performed on the discriminator. The results of more detailed ablation experiments prove that text GAN can also be based on the traditional GAN framework, and the RL is not necessary. Thus, we believe that the GAN structure that uses Gumbel-softmax approximate sampling is more flexible than the RL-based model.

2.2.4 Autoencoder Framework

Firstly, a brief introduction to the autoencoder [83] framework is given. The framework of autoencoder is shown in Figure 2.4.

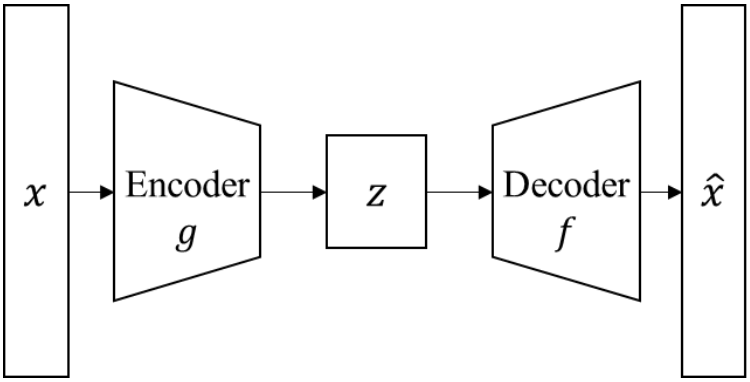

Figure 2.4 The framework of autoencoder.

The autoencoder framework consists of two major modules: the encoder and the decoder. The input data x is mapped to z in the latent space through the encoder g. Then z is mapped back to the original space through the decoder f to obtain the reconstructed sample $\hat{x}$. In the optimization process, autoencoder does not need to use the label of the sample. By minimizing the reconstruction error, the autoencoder learns

the z of the sample. This unsupervised optimization method greatly improves the model generality.

In 2018, David Donahue proposed to introduce autoencoder into GAN. The author named the model LaTextGAN [84]. The model structure of LaTextGAN is shown in Figure 2.5.

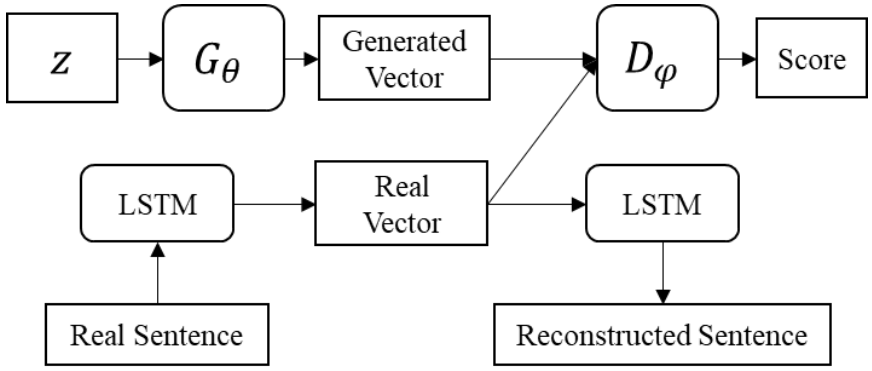

Figure 2.5 The framework of LaTextGAN.

LaTextGAN consists of three parts: a generator, a discriminator, and textual autoencoder based on LSTM. z is random noise, $score$ is the score of the input vector by the discriminato. The goal of the generator is to improve the score as much as possible. Therefore, the latent space vector generated by the generator should be as close as possible to the latent space vector obtained by the LSTM of the real sentence. This model architecture relies on the ability of the autoencoder to reproduce text, and the training process of the autoencoder and GAN must be coordinated. In addition, the model is prone to mode collapse during the training process. Limited by the extract sentence features ability of LSTM, LaTextGAN does not perform well in text generation. However, LaTextGAN is still a good model, opening up new ideas for subsequent models.

Benefiting from the idea of WGAN and the extensive use of the Transformer autoencoder, the idea of GAN in text generation can be slightly changed. The generator's output is not necessarily a sentence but also a sentence vector in the latent space. Correspondingly, the task of the discriminator has also changed. The discriminator needs to judge whether the current sentence vector is true. Therefore, TILGAN was proposed. Before TILGAN training, the author trains a Transformer-

based autoencoder on the real corpus. After the training, the sentence vectors in the real corpus through the encoder are used as real data, while the generated sentence vectors are used as fake data.

In general, there have been many excellent variants of GAN in the field of text generation in recent years. Nevertheless, there are still many problems to be solved.

Chapter 3

WRGAN: Improved RelGAN with Wasserstein Loss for Text Generation

3.1 Problem Definition

To address the discriminator's lack of ability to backpropagate the gradient to the generator, we conducted experimental testing and concluded that Gumbel-softmax relaxation technology is more effective than RL. However, since the LSTM-based generator may lack sufficient expressive power for text generation, we employed relational memory in its place. The discriminator provides insufficient guidance information to the generator. Therefore, we used the Wasserstein loss accordingly. We carefully designed the discriminator network corresponding to the Wasserstein loss to ensure the network can be coordinated. Since the improved model represents an improved RelGAN model with Wasserstein loss, we dubbed it WRGAN.

3.2 Overall Framework

The overall framework of WRGAN, as illustrated in Figure 3.1, consists of three major components: a relational memory-based generator, the Gumbel-softmax relaxation, and a 1-dimensional convolution-based discriminator. After the generator completed standard MLE training over several epochs, the network initiated adversarial training. In accordance with RelGAN, for each M_t at time t, we can obtain the updated memory $\widetilde{M}_{t+1}$ as:

$$\widetilde{M}_{t+1} = \left[\widetilde{M}_{t+1}^{1} : \ldots : \widetilde{M}_{t+1}^{H}\right], \tag{10}$$

$$\widetilde{M}_{t+1}^{h} = \sigma\left(\frac{Q_t^h {K_t^h}^T}{\sqrt{d_k}}\right) V_t^h, \tag{11}$$

where σ is softmax function. Q, K, and V denote the query, key, and value vectors, respectively. H is the number of heads. Then the output of generator o_t is given by:

$$o_t = f_\theta\left(\widetilde{M}_{t+1}, M_t\right), \tag{12}$$

where f_θ is a combination of skip connections, multi-layer perceptron (MLP), gated operations and/or pre-softmax linear transformations [85].

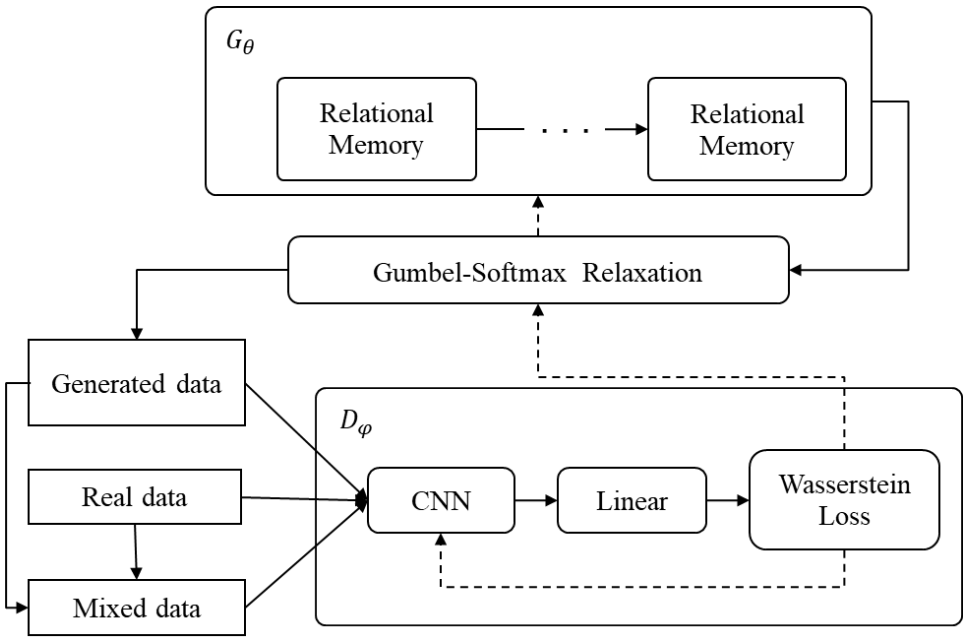

Figure 3.1 The overall framework of WRGAN.

After obtaining the generator output o_t, it is entered into Gumbel-softmax to obtain the generated data $\tilde{x}$, defined as:

$$\tilde{x} = \sigma\left(\beta(o_t + g)\right), \tag{13}$$

where σ is a softmax function. β is a tunable parameter set to 100 in this study. g is defined as:

$$g = -log\left(-log(u)\right), \tag{14}$$

where u follows a $Uniform(0,1)$ distribution.

Then, we put the generated, real, and mixed data into the discriminator to obtain the loss. This loss, which represents the relative distance between the generated and real data distributions, is used by the model to adjust network parameters.

3.3 Relational Memory-Based Generator

In this study, we introduce a powerful module in the generator in the form of relational memory [86], wherein multiple memory slots interact through Self-Attention. Relational memory module lets the generator to express the data distribution as comprehensively as possible. The structure of the generator is shown in Figure 3.2.

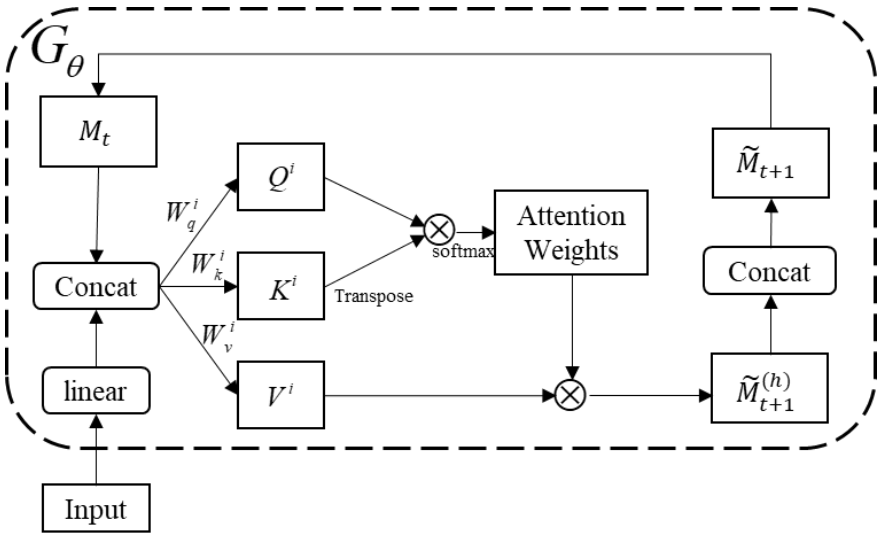

Figure 3.2 The generator framework.

In the pre-training process, the input consists of real data processed by word embedding, whereas in the adversarial training process, the input is a normally distributed random noise vector. After the linear transformation is applied, the input is merged with the memory slot of the previous moment. Subsequently, through Self-Attention and splicing operations, the memory of the current moment is formed. This process essentially enables the generator to obtain global information. As a result, the generated long text is more accurate, and the generator's expressive ability is improved.

3.4 Rebuild the Discriminator

The proposed discriminator framework is shown in Figure 3.3, and used parameters are listed in Table 3.1. We selected one-hot as the input form, and $[batchsize, vocabularysize, maximumlength]$ as the input shape. The first layer is a 1-dimensional convolutional [87] layer for dimension conversion. The second layer consists of three groups of ResBlock layers with different sizes of 1-dimensional convolution kernels. The structure of each ResBlock [88] is shown in Figure 3.4. The sizes of the three groups of convolution kernels are $[1,3,5]$, and the padding is $[0,1,2]$. The ResBlock also contains a hidden hyperparameter corresponding to its dimensionality. Different data sizes correspond to different dimensions. A detailed analysis of this parameter's impact on the model can be found in Section 3.7. The three channels are concatenated with different convolution kernel sizes. Notably, the two linear layers do not contribute any activation functions.

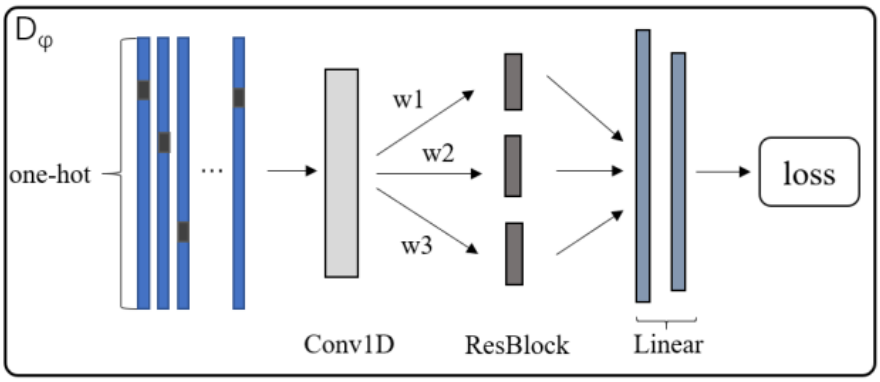

Figure 3.3 The proposed discriminator framework.

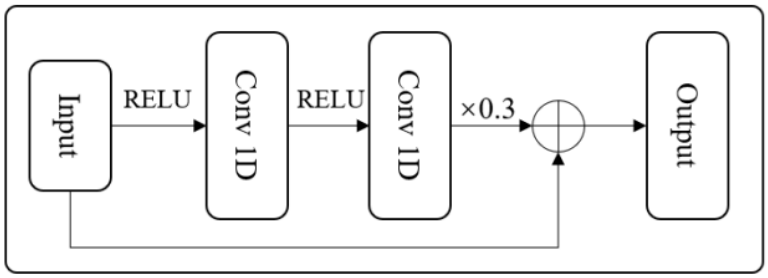

Figure 3.4 The Resblock.

Table 3.1 The discriminator parameters.

Layers	Input Shape	Kernel Shape
Conv1D	(Batch size, Vocab size, Max length)	(1, Vocab size)
ResBlock	(Batch size, Dim, Max length)	(1, Dim), (3, Dim), (5, Dim)
Linear1	(Batch size, Dim × Max length × 3)	
Linear2	(Batch size, 1000)	
Output	(Batch size, 1)	

We assume that the weight of the convolutional layer is W_d^t, where $d \in [1,2,3]$. The real and generated inputs correspond to $[r^1 : \ldots : r^T]$ and $[\hat{y}^1 : \ldots : \hat{y}^T]$, respectively. For real data, the distributed representation h_r^t is:

$$h_r^t = W_d^t r^t. \tag{15}$$

The distributed representation h_y^t of generated data is:

$$h_y^t = W_d^t \hat{y}^t. \tag{16}$$

3.5 Network Training

3.5.1 Loss Function

In this study, we employed the gradient penalty Wasserstein loss. According to WGAN-GP, the gradient penalty prevents all network parameters from approaching extreme values, ensuring that the weights are evenly distributed within a specific interval. Thus, the gradient penalty maintains the stability of adversarial training. The discriminator loss can be defined as:

$$L_D = \mathbb{E}[D(\tilde{x})] - \mathbb{E}[D(x)] + \lambda \mathbb{E}[(||\nabla_{\hat{x}} D(\hat{x})||_2 - 1)^2], \tag{17}$$

where $\hat{x}$ is mixed data by real and generated data. λ is the penalty coefficient. All experiments in this study used $\lambda = 10$. The generator loss is given by:

$$L_G = -\mathbb{E}[D(\tilde{x})]. \tag{18}$$

By plugging Equations 15 and 16 into Equations 17 and 18, the following expressions are obtained for L_D and L_G:

$$L_D = \frac{1}{3T}\sum_{d=1}^{3}\sum_{t=1}^{T} D\left(h_{\hat{y}}^{t}\right) - \frac{1}{3T}\sum_{d=1}^{3}\sum_{t=1}^{T} D(h_r^t) + \frac{\lambda}{3T}\sum_{d=1}^{3}\sum_{t=1}^{T}\left(\left|\left|\nabla_{\tilde{m}} D(h_{\tilde{m}}^{t})\right|\right|_2 - 1\right)^2, \tag{19}$$

$$L_G = -\frac{1}{3T}\sum_{d=1}^{3}\sum_{t=1}^{T} D\left(h_{\hat{y}}^{t}\right), \tag{20}$$

where $\tilde{m}$ is obtained by randomly mixing real data and generated data.

3.5.2 Training Details

Limited by hardware equipment, we set the batch size to 64 during the training process and employed the Adam [89] optimizer. During adversarial training, the learning rates of the generator and discriminator were both 1×10^{-4}. The L2 regularization weight decay was 0.01 [90], and the dropout of the discriminator was 0.25. The maximum number of iterations was 2,000, and the generator embedding [91, 92] dimension was 32.

3.5.3 Baseline

To comprehensively evaluate the proposed model, we compared its performance with that of several baselines:

(1) SeqGAN: a text GAN model based on RL algorithm and MCTS, training on COCO Image Captions, EMNLP 2017 WMT News, and Chinese poetry datasets.

(2) RankGAN: a text GAN model based on RL algorithm, training on COCO Image Captions, EMNLP 2017 WMT News, and Chinese poetry datasets.

(3) LeakGAN: a text GAN model based on RL algorithm and LSTM, training on COCO Image Captions, EMNLP 2017 WMT News, and Chinese poetry datasets.

(4) RelGAN: a text GAN model based on relational memory cell (RMC) and Gumbel-softmax, training on COCO Image Captions, EMNLP 2017 WMT News, and Chinese poetry datasets.

(5) SentiGAN: a text GAN model based on RL algorithm and LSTM, training on Movie Reviews datasets.

(6) CSGAN: a text GAN model based on RL algorithm and RNN, training on Movie Reviews datasets.

(7) CatGAN: a text GAN model based on RMC and hierarchical evolutionary learning algorithm, training on Movie Reviews datasets.

3.6 Experiments and Analysis

To evaluate the model's performance, we tested our model on real-world data, including the COCO Image Captions, EMNLP 2017 WMT News, Movie Reviews [93], and Chinese poetry datasets. Specific experimental parameter settings are presented in each corresponding subsection.

3.6.1 Evaluation Metrics

Similar to other models, we used two evaluation metrics.

The first evaluation metric is the negative log-likelihood (NLL_{gen}) [94] and its counterpart (NLL_{oracle}), defined as:

$$NLL_{gen} = -\mathbb{E}_{Y_{1:T}\sim P_r} logP_\theta(Y_1, \dots, Y_T), \tag{21}$$

$$NLL_{oracle} = -\mathbb{E}_{y_{1:T}\sim P_\theta} logP_r(y_1, \dots, y_T), \tag{22}$$

where P_θ and P_r are the generated and real data distributions, respectively. We used NLL_{gen} to evaluate the diversity of generated data.

The other evaluation metrics of real data is bilingual evaluation understudy (BLEU). The BLEU score is used to compare and count the number of commonly occurring $n-gram$ words for the quality evaluation of the generated text. To enable BLEU scores as an evaluation metric, we also use test data [95].

3.6.2 COCO Image Captions

The Microsoft COCO dataset consists of 20,000 manually generated image captions. After preprocessing [96], we got a dictionary with 4,682 unique words, and the maximum sentence length is 37. Additional properties of this dataset are listed in Table 3.2.

Table 3.2 The details of the COCO Image Captions dataset.

Vocabulary size	4,682
Maximum length	37
Number of training sentences	10,000
Number of test sentences	10,000

A comparison between the BLEU scores of tested models is presented in Table 3.3. We adopted the same evaluation settings for all models. Except for the BLEU-5 score, all scores achieved by our model exceeded those produced by other models. This indicates that our model is highly effective for the COCO dataset. The NLL_{gen} score shows that the improved model also performed well on data diversity.

Figure 3.5 shows a line graph generated after 2,000 training iterations for a hyperparameter dimension of 128. No strong fluctuations were observed in the training process, indicating relative stability in the model. The samples generated from the COCO dataset can be found in Section 3.8.1.

Table 3.3 The BLEU and NLL_{gen} scores on COCO Image Captions dataset.

Method	BLEU-2	BLEU-3	BLEU-4	BLEU-5	NLL_{gen}
MLE	0.731	0.497	0.305	0.189	0.718
SeqGAN	0.745	0.498	0.294	0.180	1.082
RankGAN	0.743	0.467	0.264	0.156	1.344
LeakGAN	0.746	0.528	0.355	0.230	0.679
RelGAN (100)	0.849	0.687	0.502	0.331	0.756
RelGAN (1000)	0.814	0.634	0.455	0.303	0.655

WRGAN	0.853	0.687	0.509	0.318	0.673

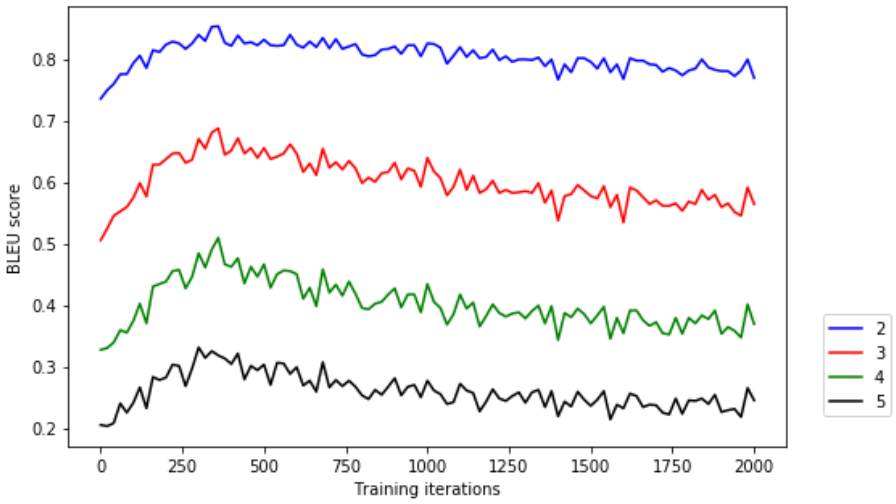

Figure 3.5 The BLEU scores for COCO Image Captions.

3.6.3 EMNLP 2017 WMT News

Table 3.5 lists the BLEU and NLL_{gen} scores for the EMNLP 2017 WMT News dataset, with the dimension of 256. The dataset in question contains approximately 270,000 sentences, as well as 10,000 sentences of testing data. After preprocessing, we obtained a vocabulary size of 5,255, with a maximum sequence length of 51. The additional details about the dataset are listed in Table 3.4.

Table 3.4 The details of the EMNLP 2017 WMT News dataset.

Vocabulary size	5,255
Maximum length	51
Number of training sentences	270,000
Number of test sentences	10,000

The training curves corresponding to BLEU scores are shown in Figure 3.6. These results indicate that, despite slightly weaker data diversity, the improved model outperformed all other models in terms of BLEU scores. The figure also shows that the model can achieve higher performance and more stable results even on a large-scale

dataset. Data samples generated from the EMNLP 2017 WMT News dataset can be found in Section 3.8.2.

Table 3.5 The BLEU and NLL_{gen} scores on EMNLP 2017 WMT News dataset.

Method	BLEU-2	BLEU-3	BLEU-4	BLEU-5	NLL_{gen}
MLE	0.768	0.473	0.240	0.126	2.382
SeqGAN	0.777	0.491	0.261	0.138	2.773
RankGAN	0.727	0.435	0.209	0.101	3.345
LeakGAN	0.826	0.645	0.437	0.272	2.356
RelGAN(100)	0.881	0.705	0.501	0.319	2.482
RelGAN(1000)	0.837	0.654	0.435	0.265	2.285
WRGAN	0.952	0.782	0.539	0.336	2.812

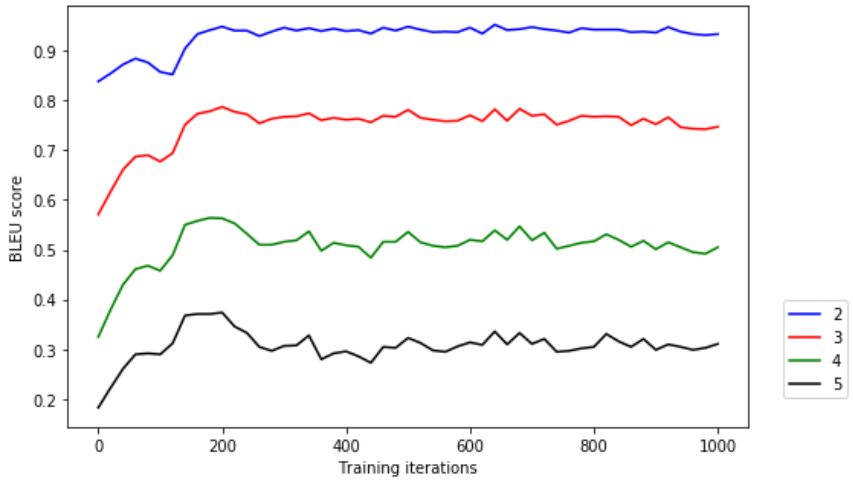

Figure 3.6 The BLEU scores for EMNLP 2017 WMT.

3.6.4 Chinese Poetry

The Chinese poetry dataset consists of 16,394 Tang poems, each of which contains five Chinese words per sentence. Additional details pertaining to this dataset are presented in Table 3.6.

Table 3.6 The details of the Chinese Poetry dataset.

Vocabulary size	4,140
Maximum length	20
Number of training sentences	8,197
Number of test sentences	8,197

Since we used the BLEU score as the evaluation metric, we randomly selected 8,197 poems as the training set, with the remaining 8,197 poems allocated to the testing set. After preprocessing, we obtained a vocabulary with a size of 4,140. Table 3.7 lists the BLEU-2 scores for this dataset. Here, the improved model also achieved satisfactory results.

Table 3.7 The BLEU-2 scores on the Chinese poetry dataset.

Method	SeqGAN	RankGAN	RelGAN	LeakGAN	WRGAN
BLEU-2	0.738	0.812	0.817	0.456	0.835

3.6.5 Movie Reviews (MR)

The Movie Reviews (MR) dataset contains two sentiment classes (negative and positive) and 4,503 total samples. Of these samples, 3,152 were allocated for training, with the remaining 1,351 set apart for testing. Additional details pertaining to this dataset are listed in Table 3.8.

Table 3.8 The details of the Movie Reviews dataset.

Vocabulary size	6,216
Maximum length	15
Number of training sentences	3,152
Number of test sentences	1,351

After preprocessing, we obtained a vocabulary with a size of 6,216 and a maximum sentence length of 15. Table 3.9 lists the BLEU and NLL scores for Movie Reviews dataset. Except BLEU-5, all scores obtained by the improved model were satisfactory.

Table 3.9 The BLEU and NLL_{gen} scores for the Movie Reviews dataset.

Method	SentiGAN	CSGAN	CatGAN [97]	WRGAN
BLEU-2	0.532	0.452	0.589	0.623
BLEU-3	0.285	0.204	0.335	0.337
BLEU-4	0.167	0.112	0.194	0.193
BLEU-5	0.143	0.082	0.144	0.128
NLL_{gen}	2.436	2.912	1.619	0.8061

3.6.6 Comparison of RelGAN and WRGAN on COCO Dataset

The discriminator loss when using the Wasserstein loss in RelGAN is shown in Figure 3.7. We found that the discriminator and generator losses were nearly equal to zero throughout the training process, ranging from 0.0005 to 0.005. In this case, the discriminator yields no useful guidance information to the generator, and the two networks cannot perform adversarial learning. Therefore, the discriminator did not perform well on these datasets. Excluding some obvious causes of error, we located the problem in the discriminator structure, which was modified after several experiments.

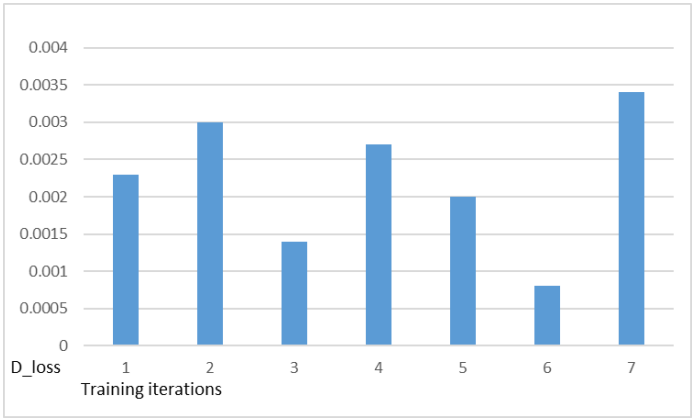

Figure 3.7 The discriminator loss of RelGAN.

Figure 3.8 compares the BLEU-2 scores of the three RelGAN-based models based on the COCO dataset with 1,400 iterations. The parameters in the figure represent those used in the original study. We found RelGAN to exhibit strong fluctuations when using the Wasserstein loss, which resulted in suboptimal performance.

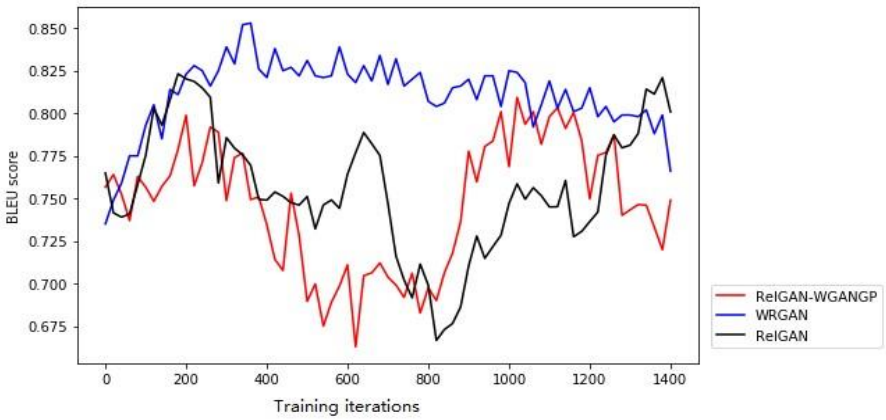

Figure 3.8 The BLEU-2 scores on the COCO Image Captions compared with RelGAN.

3.7 Impact of Hyperparameters

3.7.1 Impact of Dimension

This section analyzes the impact of the dimension hyperparameter on the model. In Figure 3.9, the BLEU-2 scores of the model on the COCO dataset are plotted for dimensionalities of 128 and 64. For the dimensionality of 128, overfitting is apparent after approximately 1,000 iterations. However, when the number of dimensions was 64, there was no clear overfitting. After the experiments, we determined that the dimensionality should be proportional to the amount of training data. The training curve for the EMNLP 2017 WMT News dataset confirms this view, as it did not exhibit obvious overfitting with a dimensionality of 256. The dimension hyperparameter was not minimized, as the model may not fully obtain the sentence vector's characteristics with a small dimensionality. Accordingly, we ensured that the dimensionality was sufficiently large.

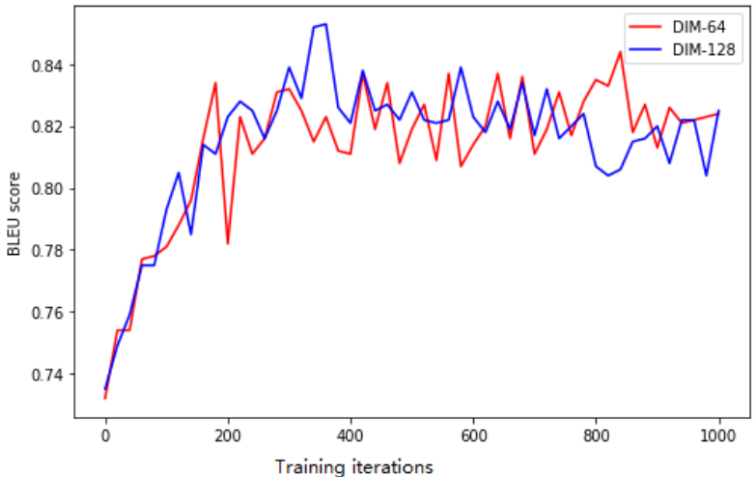

Figure 3.9 The BLEU-2 scores on COCO Image Captions where the dimension is 64 and 128.

3.7.2 Impact of k

The hyperparameter k represents the quotient of the generator and discriminator steps. As shown in Figure 3.10, the model gradually deteriorated and destabilized as k increased. We believe that this was caused by an insufficient amount of discriminator training, as the generator became too "experienced" to perform adversarial training. Perhaps, as the number of training sessions increases, the model will gradually stabilize again. However, this remains conjecture, and k cannot be minimized. A smaller value of k corresponds to increased training time for the discriminator, which in turn accelerates the overfit. Therefore, after several experiments, we set k to 1/5.

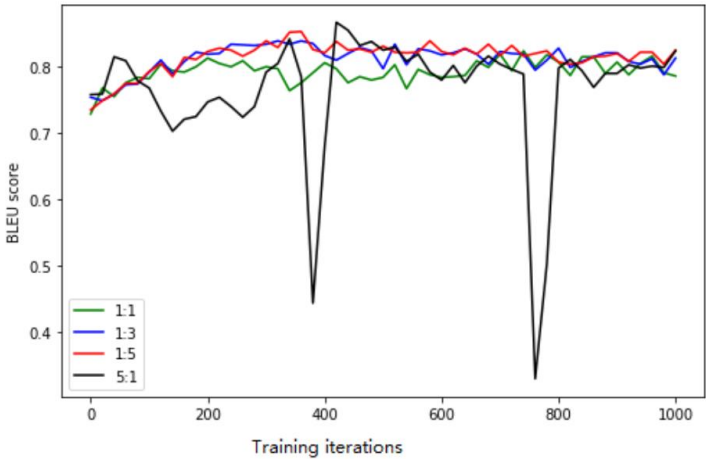

Figure 3.10 The BLEU-2 scores on COCO Image Captions where k is 5, 1, 1/3 and 1/5.

3.8 Case Study

We randomly selected 10 generated samples from the short and long text datasets for this case study. The maximum lengths of the generated sentences in the two datasets were 15 and 51. The performance of the model on the two datasets exhibited good agreement with experimental data. In other words, owing to the relatively simple distribution of samples in the short text dataset, the model achieved higher accuracy but poor diversity. In contrast, the generated sentences corresponding to the long text dataset featured lower precision but higher diversity.

3.8.1 The Generation Data from COCO Image Captions

Table 3.10 lists samples generated during adversarial training with a dimension of 64 and k of 1/5, after pre-training with the COCO image caption dataset. The generated sentences do not have obvious errors, such as the identifiers “EOS” and “BOS.” However, the samples do exhibit minor grammatical errors. The accuracy of sentences generated by the model on the short-text dataset is therefore acceptable. We found that the majority of sentences start with the article “a,” which indicates that sentence diversity must be improved.

Table 3.10 The samples generated from the COCO image caption dataset.

Samples
a yellow bicycle parked next to a red wall.
a white and blue plane flying in the blue sky.
a white cat has caught a bird on its tail.
a woman in the kitchen is holding a dog.
a man is looking at motorcycle in the road by the building.
two wet young boys on a table.
a bathroom with a toilet and a sink.
a young girl sitting on top of a bike near the ocean.

some people on the snow covered field.

many motorcyclists gather in front of a bus.

3.8.2 The Generation Data from EMNLP 2017 WMT News

Table 3.11 lists samples generated during adversarial training from the EMNLP 2017 WMT News dataset for the dimensionality of 256 and k of 1/5. The sentences generated here do not exhibit obvious errors, such as the identifiers "EOS" and "BOS." However, these samples show more grammatical errors than those generated with the short text dataset. This may be because the model is not sufficiently complex to accurately express the distribution of real samples. In terms of the diversity of generated sentences, the model performed relatively well without many repeated words.

Table 3.11 The samples generated from EMNLP 2017 WMT News dataset.

Samples
the security services, an independent has already said: "there is a need to follow up the process of the project in 2017.
i have been together with russia, because we just don't know if that is the need to make it," he said.
he will tell the player he didn't want to continue to make his opinion on that.
we are making sure that we will better understand that we need to strengthen our order for the next 18—and we will get ourselves into our future.
in some cases, it is the first time in the past three years, a few of them needed to be with other. 1% of the population is even more within the trump administration.
after the festival, his wife, who was in contact with a police in the UK.
now, it is amazing when people on the court, including a man from the police.
"I'm really concerned, because it was a very careful in my life," she said.
he has had to quit from the hospital after a 13-year-old who had been on three years.
"we are back with this, and we are not about that," he said in a statement for several years.

3.9 Summary

In this study, we proposed a new and improved model for text generation based on RelGAN and WGAN-GP and named it WRGAN. We redesigned the model structure to allow various modules to operate in coordination, applied the Wasserstein Distance in text generation to provide more useful information to the generator, and employed relational memory as the generator architecture to reduce mode collapse. Compared with existing models, our model produced higher evaluation scores and sample quality. The proposed model also achieved superior results in a comparative experiment using RelGAN. We then analyzed the effect of hyperparameters on model performance subsequently, tested the model using multiple real datasets. Several important issues remain open, including relatively low BLEU-5 scores and suboptimal performance with synthetic data. However, we still need to adjust the model structure to address these issues. We plan to continue improving this network in the future and apply it to further NLP tasks.

Chapter 4

TSGAN: Improved Transformer-Based Implicit Latent GAN with Multi-head Self-Attention

4.1 Introduction

In this study, we propose a new generator architecture based on the Multi-head Self-Attention and linear layers. The overall structure of GAN is improved from TILGAN. We rebuilt the generator architecture with Multi-head Self-Attention to make the generator obtain superior text generation capabilities. Our model consists of a Transformer autoencoder, a generator with Multi-head Self-Attention, and a linear discriminator. We use the Kullback-Leibler (KL) divergence as the GAN's loss functions. The encoder of the Transformer autoencoder encodes the distribution of real samples, and the decoder decodes real sample encoding or generated sample encoding into text. The loss function of autoencoder is cross entropy. The detailed model structure and parameters can be found in section 4.2. We experiment on the MSCOCO and EMNLP WMT News datasets. The proposed model has achieved a better BLEU score than TILGAN. Through the ablation experiments, we prove that the proposed generator has a better ability for unconditional text generation.

4.2 Overall Framework

The overall framework of our model is shown in Figure 4.1. The model receives a random noise ε under a Gaussian distribution and takes in real text samples from a corpus X. Through the generator network G_θ, random noise ε is transformed into the generated sentence vector $\hat{Z}$. Through the Transformer encoder, the real text sample is transformed into Z. $\hat{X}$and $\tilde{X}$ represent the sentences obtained by $\hat{Z}$ and Z through the Transformer decoder, respectively.

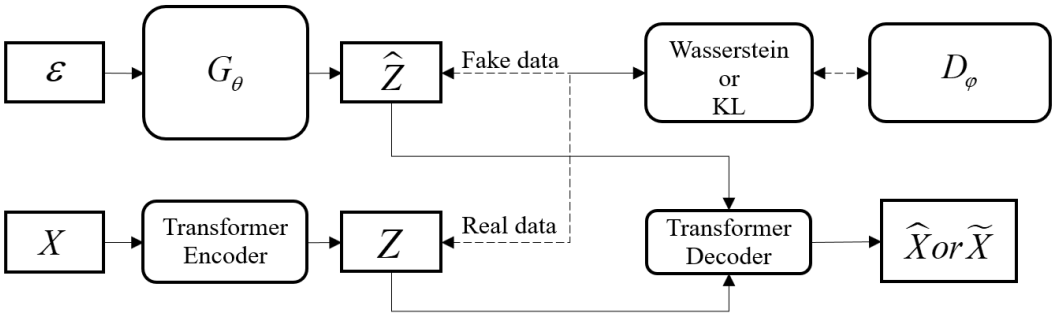

Figure 4.1 Overall framework of TSGAN.

The proposed GAN framework can be divided into three parts: the Transformer autoencoder, the generator and the discriminator.

The Transformer encoder is used to generate the distribution of real samples, and the decoder is used to decode sentence vector into text. The loss function of autoencoder is cross entropy. The task of Transformer is to minimize the gap between X and $\tilde{X}$ to ensure the accuracy of the real sentence vector distributions.

The discriminator consists of three linear layers and two BN layers, with the ReLU activation function for each layer. The loss function of GAN is Wasserstein Distance or KL divergence. The generator's goal is to minimize the distance between the generated sentence vector and the sentence vector of the real sample. On the other hand, the discriminator tries to maximize the distance between real and fake data.

4.3 Multi-head Self-Attention-Based Generator

Different from the stacking of linear layers and BN layers of the TILGAN generator, we use Multi-head Self-Attention to build the generator. The proposed generator framework is shown in Figure 4.2, where $\otimes$ means the dot product, and the two linear layers are used for reshaping.

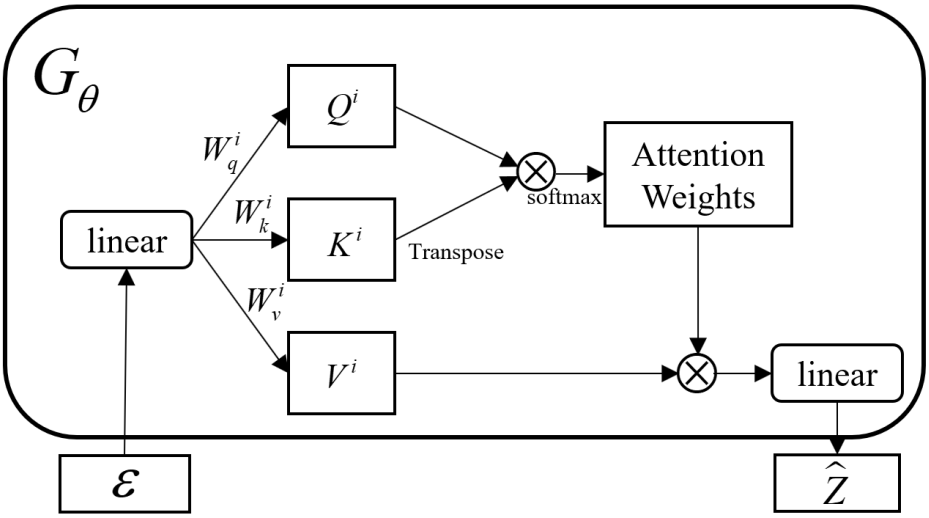

Figure 4.2 Proposed generator framework.

4.3.1 Multi-head Self-Attention

In recent years, the Self-Attention has been widely used in various deep-learning-based NLP tasks. It learns the similarity between each source word and the other words. Self-Attention can capture syntactic or semantic features between words in the same sentence. In essence, the Self-Attention maps the query and a series of key-value pairs to an output.

The Multi-head Self-Attention performs multiple sets of Self-Attention on the original input sequence. Then, each set of Self-Attention results is concatenated and linearly transformed to obtain the final output. Multi-head Self-Attention utilizes multiple versions of the same query to implement multiple the Attention modules in parallel. Each head has its multiple query, key, and value vectors. It obtains multiple queries by linearly transforming the query through different weight matrices.

4.3.2 The Proposed Generator

The essence of the Self-Attention is to obtain the attention weight of each position of the sentence in the encoding process through a mathematical calculation and then calculate the implicit vector representation of the entire sentence in the form of the weighted sum. We note that the essence of the Self-Attention coincides with the generative process of GAN. GAN also generates complex distributed data through a series of transformation calculations. We believe that the Self-Attention and GAN can work well together. Therefore, we add the Multi-head Self-Attention to the GAN generator. The influence of the hyperparameter head number $nhead$ on the model and the choice of the $nhead$ are described in detail in section 4.8.2.

As shown, the random noise input goes through a linear layer. The primary role of this layer is to reshape noise. After the weighted calculation of Multi-head Self-Attention, the sentence vector is generated through another linear layer. Formally, we employ $L(\varepsilon)$ to represent the processed noise ε through the linear layer. By setting the number of Attention heads to I, we can get I sets of queries, keys, and values. For each Attention head, we have:

$$\begin{cases} Q^i = L(\varepsilon)W_q^i \\ K^i = L(\varepsilon)W_k^i \\ V^i = L(\varepsilon)W_v^i. \end{cases} \tag{23}$$

Accordingly, we can get the sentence vector $\hat{Z}$ by:

$$\hat{Z} = L'\left(\sigma\left(\frac{Q^i(K^i)^T}{\sqrt{d_k}}\right)V^i\right), \tag{24}$$

where σ is the softmax function, and d_k is the column dimension of the keys, L' is the linear layer after Muti-head Self-Attention.

4.4 The Proposed Discriminator

The proposed discriminator framework is shown in Figure 4.3. The $Input$ is a mixture of the generated sentence vector and the sentence vector obtained from the real sample using a Transformer autoencoder. The mixed input passes through the

discriminator's three linear layers, two BN layers, and two RELU layers to get the current batch score. The score, in turn, guides the output of the generator. The input and output shapes of the three linear layers are listed in Table 4.1.

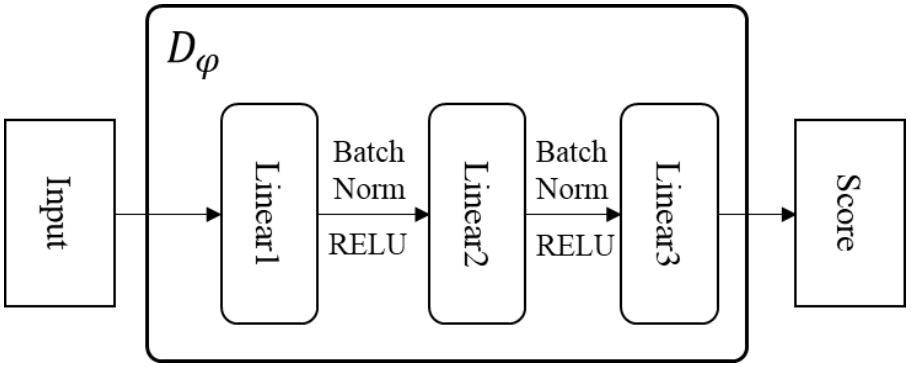

Figure 4.3 The proposed discriminator framework.

Table 4.1 The discriminator parameters.

Layers	Input Shape	Output Shape
Linear1	(Batch size, Sentence vector shape)	(Batch size, 300)
Linear2	(Batch size, 300)	(Batch size, 300)
Linear3	(Batch size, 300)	(Batch size, 1)

4.5 The Transformer Autoencoder

The Transformer autoencoder framework is shown in Figure 4.4. The role of Transformer autoencoder in the model is divided into two parts. The encoder encodes the input real samples into sentence vectors as positive samples and inputs them into the discriminator, while the decoder decodes the sentence vectors generated by the generator into readable sentences in the test and evaluation.

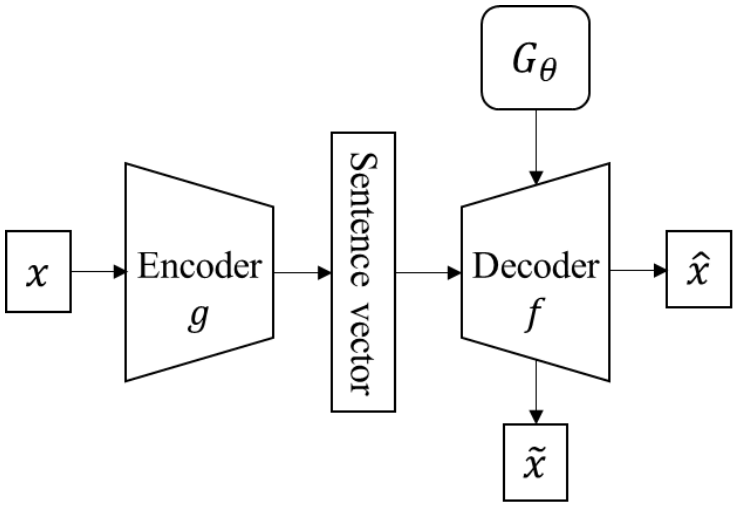

Figure 4.4 The Transformer autoencoder framework.

During the training process, the task of the Transformer autoencoder is to restore the input samples. $\hat{x}$ should be as similar to x as possible. In this study, we use cross entropy as the loss function of the autoencoder. Since the autoencoder directly affects the generation, we should choose the autoencoder parameters carefully. In section 4.8.2, we detail the impact of the choice of the learning rate of the autoencoder on model generation. The best autoencoder parameter choices so far are: 2 layers, 4 heads, 512 hidden dimensions, and a 0.12 learning rate.

4.6 Network Training

4.6.1 Loss Function

As mentioned above, autoencoder uses cross entropy as loss function.The cross entropy can be defined as:

$$L = -\sum_{i}^{n} p(x_i) log(q(x_i)) \tag{25}$$

GAN can use Wasserstein Distance or KL divergence as loss function. Since the Wasserstein Distance has some additional hyperparameters, the current loss function is temporarily KL divergence. KL divergence can be defined as:

$$D_{KL}(P||Q) = -\sum_{i}^{n} P(i) ln \frac{Q(i)}{P(i)}. \tag{26}$$

4.6.2 Training Details

Limited by hardware equipment, we set the batch size to 64 or 128. We used the Adam optimizer, the learning rate of the generator is 1×10^{-4}. The learning rate of the discriminator is 1×10^{-4}, and the learning rate of autoencoder is 0.12.

4.6.3 Baseline

To comprehensively evaluate the proposed model TSGAN, we compare our model with the following baselines:

(1) SeqGAN: a text GAN model based on RL algorithm and MCTS, training on MSCOCO dataset, EMNLP WMT News dataset.

(2) RankGAN: a text GAN model based on RL algorithm, training on MSCOCO dataset, EMNLP WMT News dataset.

(3) LeakGAN: a text GAN model based on RL algorithm and LSTM, training on MSCOCO dataset, EMNLP WMT News dataset.

(4) GSGAN [98]: a text GAN model based on Gumbel-softmax and LSTM, training on MSCOCO dataset, EMNLP WMT News dataset.

(5) WGAN: a GAN model based on CNN and Wasserstein Distance, training on MSCOCO dataset, EMNLP WMT News dataset.

(6) TILGAN: a text GAN model based on Transformer autoencoder, training on MSCOCO dataset, EMNLP WMT News dataset.

4.7 Experiments and Analysis

To evaluate the performance of the model, we tested our model on real-world data, including the MSCOCO and EMNLP WMT News datasets. The specific experimental parameter settings are given in each subsection.

4.7.1 Evaluation Metrics

In this study, two metrics were used to evaluate the model. The first metric is the bilingual evaluation under study (BLEU-test). This score indicates the similarity of the candidate sentence to the reference sentence. The BLEU-test value is in the range [0,1], and a larger BLEU-test value indicates a better generation result. The BLEU score can provide an overall assessment of model quality. The BLEU-test can be defined as:

$$BLEU = BP \times exp\left(\sum_{n=1}^{N} w_n log p_n\right). \tag{27}$$

where w_n is the weight of the $n-gram$. Usually, $w_n = 1/N$.

BP is the penalty coefficient. BP can be defined as:

$$BP = \begin{cases} 1 & if\ c > r \\ e^{(1-r/c)} & if\ c \le r, \end{cases} \tag{28}$$

where c is the length of the candidate sentence, r is the length of the valid reference sentence.

p_n can be defined as:

$$p_n = \frac{\sum_{C \in M} \sum_{m \in C} Count_{clip}(m)}{\sum_{C' \in M} \sum_{m' \in C'} Count(m')}, \tag{29}$$

where M is the set of candidate sentences, m and m' are the $n-gram$ of candidate sentences C and C', respectively. $Count$ is the number of times that the $n-gram$ appears in candidate sentences. $Count_{clip}(m)$ can be defined as:

$$Count_{clip}(m) = min(Count(m),\ Count_{ref}(m)), \tag{30}$$

where $Count_{ref}$ is the number of times that the $n-gram$ appears in reference sentences.

The second metric is Self-BLEU. Self-BLEU is a diversity metric that calculates the similarity between a generated sentence and the entire remaining generation. A lower Self-BLEU score indicates a higher diversity in the generated texts.

Specifically, following Chen [99] et al., we report BLEU-2, 3, 4, 5 for BLEU-test and BLEU-2, 3, 4 for Self-BLEU.

4.7.2 Microsoft Common Objects in Context (MSCOCO)

In order to test our model, we first conduct experiments on the MSCOCO dataset. All the preprocessing steps are the same as other models. The details of the dataset are listed in Table 4.2.

Table 4.2 The details of MSCOCO dataset.

Vocabulary size	27,842
Average length	10.4
Number of training sentences	120,000
Number of test sentences	10,000

We set the Transformer autoencoder with 2 layers, 4 heads, and 512 hidden dimensions. In addition, we set the generator with 4 heads, a 256 head size, and 32 hidden dimensions. All the sentences will be padded to the maximum length during training. Then the BLEU scores on the MSCOCO dataset are listed in Table 4.3. The proposed model has achieved significantly better performance than the existing models in BLEU-2, 3, and 4 and Self-BLEU-2, and 3. The results suggest that our text generation model is generally more effective on the MSCOCO dataset than the existing models.

Table 4.3 The BLEU scores on MSCOCO dataset.

Method	BLEU-test				Self-BLEU		
	BLEU-2	BLEU-3	BLEU-4	BLEU-5	BLEU-2	BLEU-3	BLEU-4
SEQGAN	0.820	0.604	0.361	0.211	0.807	0.577	0.278
RANKGAN	0.852	0.637	0.389	0.248	0.822	0.592	0.288
LEAKGAN	0.922	0.797	0.602	0.416	0.912	0.825	0.689
GSGAN	0.810	0.566	0.335	0.197	0.785	0.522	0.230
WGAN	0.730	0.538	0.342	0.125	0.904	0.809	0.690
TILGAN	0.967	0.903	0.772	0.532	0.616	0.356	0.099
Our model	0.986	0.928	0.799	0.420	0.548	0.270	0.121

4.7.3 EMNLP WMT News

We also conduct experiments on the EMNLP WMT News dataset. All the preprocessing steps are the same as other models. The details of the dataset are listed in Table 4.4.

Table 4.4 The details of EMNLP WMT News dataset.

Vocabulary size	5,728
Average length	27.8
Number of training sentences	278,000
Number of test sentences	10,000

We set the Transformer autoencoder [100] with 2 layers, 4 heads, and 512 hidden dimensions. In addition, we set the generator with 2 heads, a 256 head size, 32 hidden dimensions, and 128 batch size. All the sentences will be padded to the maximum length during training. The model is iterated 100 times on the EMNLP WMT News dataset. The curve changes of the BLEU-test and Self-BLEU scores of the model are shown in Figure 4.5 and Figure 4.6, respectively. For BLEU-test, the higher the model score can reflect the higher quality of the generated text to a certain extent so that the BLEU-test score should be as high as possible. For Self-BLEU, the lower the model score, the lower the repetition of the generated text is, and the more diverse the generated texts are. Therefore the score of Self-BLEU should be as low as possible. As a result, the training process is to reduce the Self-BLEU score as much as possible while ensuring a high BLEU-test score to achieve the optimal solution of the model.

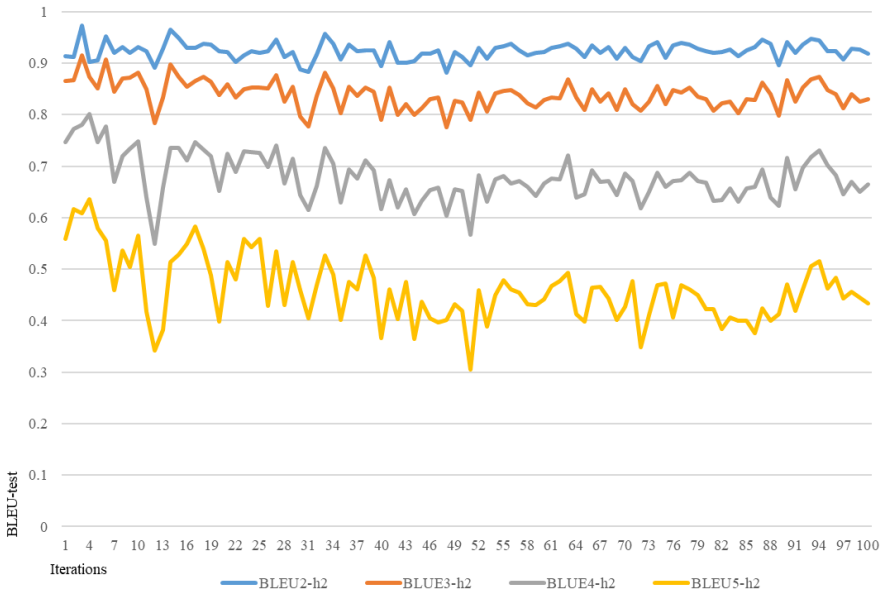

Figure 4.5 Training curves of BLEU-test scores on EMNLP WMT News dataset.

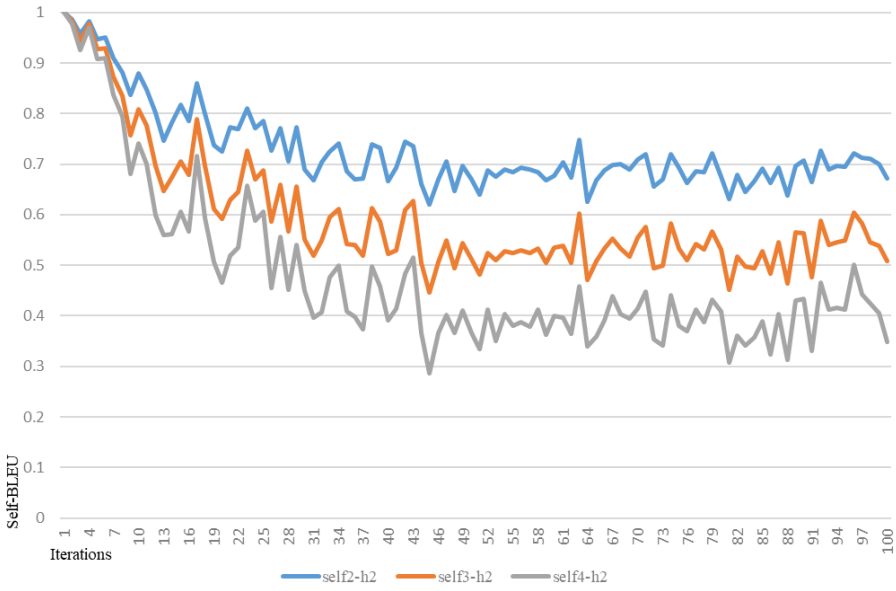

Figure 4.6 Training curves of Self-BLEU scores on EMNLP WMT News dataset.

Then the BLEU scores on the WMT News dataset are listed in Table 4.5. The proposed model has achieved significantly better performance than the existing models in BLEU-2, 3, and 4 and Self-BLEU-2. The results suggest that our text generation model is generally more effective on the EMNLP WMT News dataset than the existing models.

Table 4.5 The BLEU scores on EMNLP WMT News dataset.

Method	BLEU-test				Self-BLEU		
	BLEU-2	BLEU-3	BLEU-4	BLEU-5	BLEU-2	BLEU-3	BLEU-4
SEQGAN	0.630	0.354	0.164	0.870	0.728	0.411	0.139
RANKGAN	0.774	0.484	0.249	0.131	0.672	0.346	0.118
LEAKGAN	0.920	0.725	0.502	0.321	0.857	0.696	0.373
GSGAN	0.723	0.440	0.210	0.107	0.682	0.410	0.231
WGAN	0.891	0.774	0.502	0.267	0.933	0.910	0.886
TILGAN	0.929	0.817	0.617	0.407	0.663	0.445	0.280
Our model	0.937	0.840	0.640	0.400	0.638	0.464	0.312

4.7.4 Ablation Experiment

To indicate that our changes to the generator are effective, we also conduct an ablation experiment on the MSCOCO dataset. We keep all model parameters the same with TILGAN except the generator (including learning rate, model structure, number of autoencoder layers, and number of hidden layers). The BLEU-3 curve is shown in Figure 4.7. The overfitting part is not shown in the figure.

The curves show that our generator converges much faster than TILGAN. The results show that our generator has better text generation ability. Our model can achieve better results on large datasets and for long text generation than the original generator.

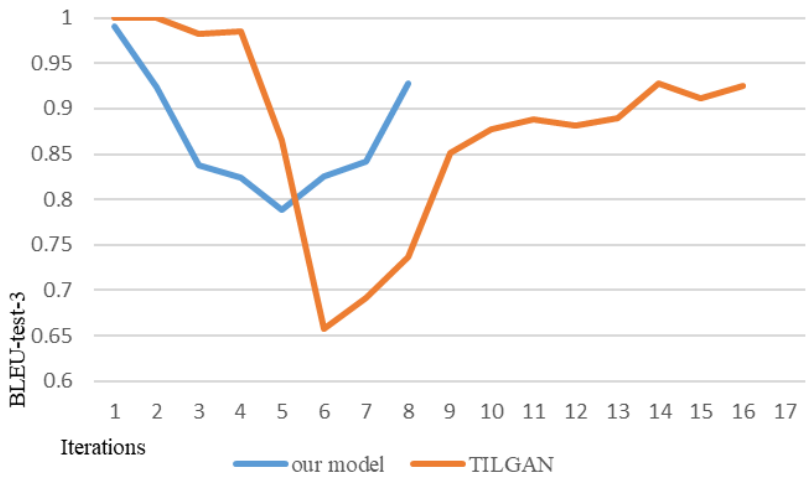

Figure 4.7 Ablation Experiment.

4.8 Impact of Hyperparameters

The model inevitably contains hyperparameters. The appropriateness of the hyperparameters also affects the performance of the model. In this section, we select three representative hyperparameters for discussion: head number $nhead$, learning rate lr of the Transformer autoencoder, and initial distribution of the input noise. Notably, the single-variable principle is used in all hyperparameter experiments.

4.8.1 Impact of the Head Number

This study uses the Multi-head Self-Attention mechanism to enhance the generator's performance. As a result, the number of heads significantly impacts the model's performance and the diversity of generated data. After 100 iterations on the EMNLP WMT News dataset, the performance of the model when $nhead$ is 2 and 4 is shown in Figure 4.8, where $BLEUx-h2\ (x=2,3,4,5)$ represents the corresponding BLEU scores when $nhead$ is 2. Correspondingly, $BLEUx-h4$ represents the BLEU scores when $nhead$ is 4.

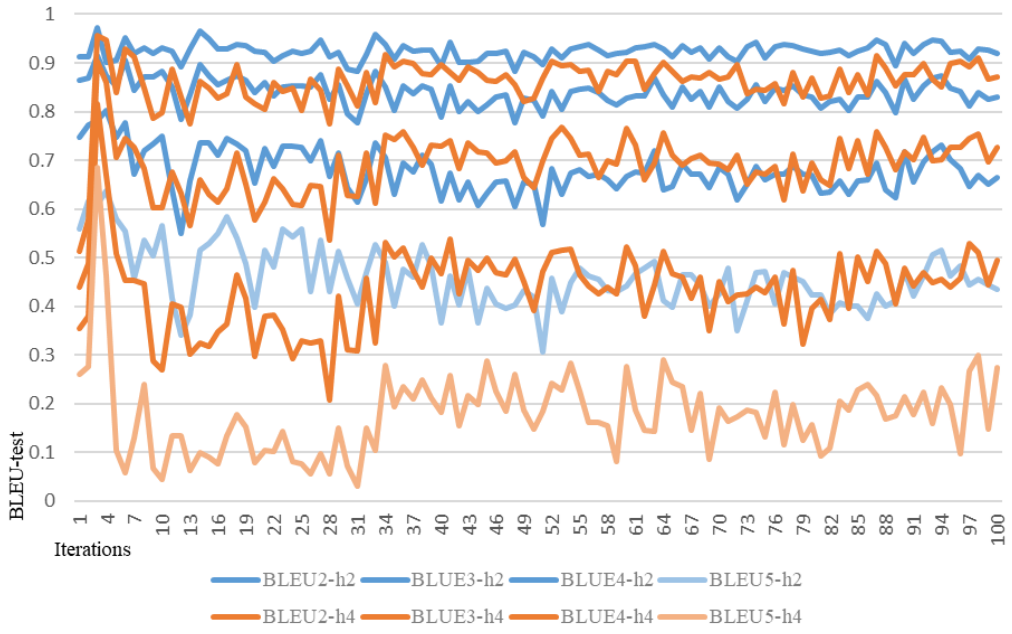

Figure 4.8 Training curves of BLEU-test scores on EMNLP WMT News dataset with different $nhead$.

From the figure, we find that the scores of BLEU-4,5 of the model when $nhead$ is 4 are much lower than that when $nhead$ is 2, whereas the scores of BLEU-2,3 are relatively close for the same scenarios. This result shows that the sentences generated when $nhead$ is 4 are trivial and of relatively poor quality. In addition, the data of the first ten iterations suggest that there is another possibility that the model may overfit. This is also one of the reasons why we finally chose 2 heads. Subsequently, we obtained the Self-BLEU score map of the model iterated 100 times on the dataset.

The Self-BLEU score curve of the model iterating 100 times on the dataset is shown in Figure 4.9. The three blue curves represent the scores of Self-BLEU-2, 3, and 4 when $nhead$ is 2. Correspondingly, the three orange curves represent the scores of Self-BLEU-2, 3, and 4 when $nhead$ is 4. Since the lower the Self-BLEU score, the better, we find that the model with 4 heads outperforms that with 2 heads on the diversity of the generated text. It again shows that the data generated by the model when the $nhead$ is 4 is scattered. In contrast, perhaps because of the characteristics of the Multi-head Self-Attention mechanism, each head pays attention to different aspects of the data, making the generated text more diverse.

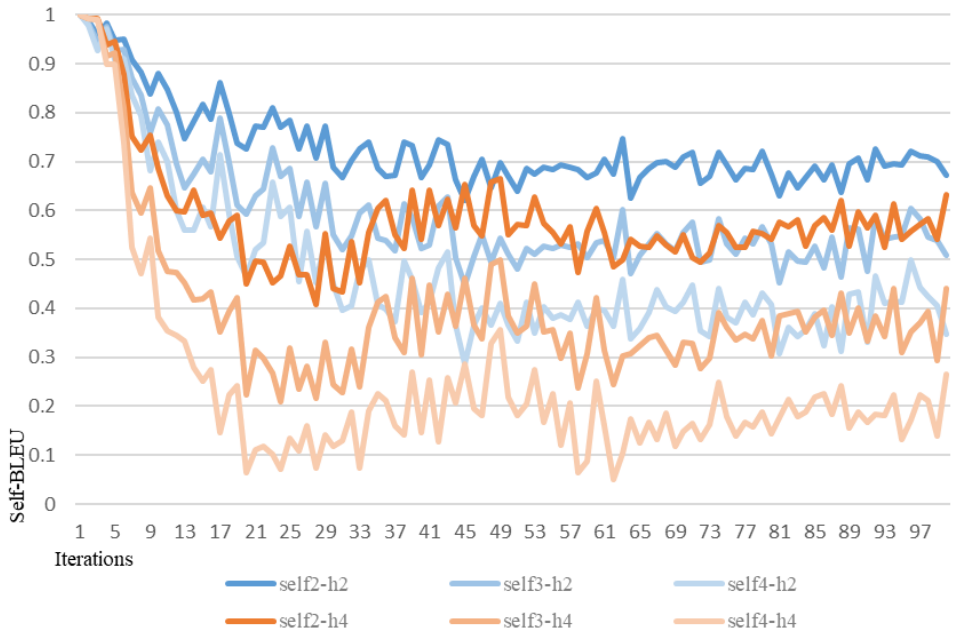

Figure 4.9 Training curves of Self-BLEU scores on EMNLP WMT News dataset with different $nhead$.

In addition, the experimental results of Figure 4.8 and Figure 4.9 also confirm an inversed relationship between the quality and diversity of the generated samples. Therefore, during the training process, we must find a delicate balance between the generation quality and diversity of the model. Thus, the generated samples have both excellent quality and diversity. Subsequently, we also conducted experiments with 1 head. When the $nhead$ is 1, the model fails to converge. We believe the model is not sufficiently complex to express the latent space distribution of the current dataset when the $nhead$ is 1. We will continue to explore this question in future research.

4.8.2 Impact of the Learning Rate

Adjustment of learning rate is a significant part of parameter adjustment. The learning rate is one of the the essential hyperparameters. Our model had the largest effective capacity when the learning rate was optimal. Therefore, to train a neural network, one of the critical hyperparameters that must be set is the learning rate [101]. Choosing the optimal learning rate is essential because it determines whether the neural network can converge to a global minimum. Larger or smaller learning rates will trap the model at the saddle point. We determined the current optimal learning rate on the EMNLP WMT News dataset by gradually increasing the learning rate lr of the Transformer autoencoder for each experiment. When the learning rate lr of the autoencoder is 0.12 and 0.15, respectively, the BLEU-test and Self-BLEU scores of the model are shown in Figure 4.10 and Figure 4.11, respectively. The blue curve corresponds to the BLEU-test and Self-BLEU score curves of the model when lr is 0.12. Correspondingly, the orange curve represents the BLEU-test and Self-BLEU score curves of the model when lr is 0.15

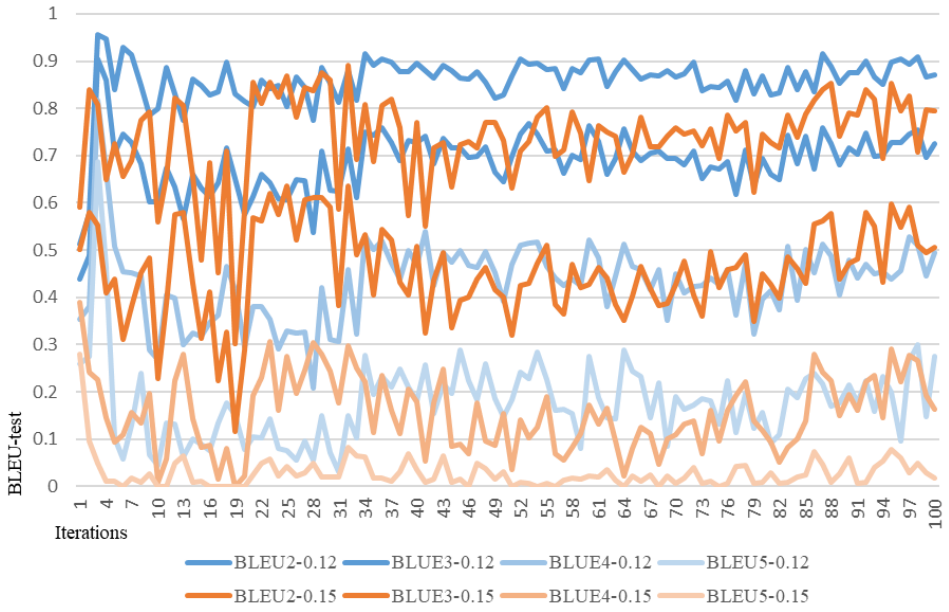

Figure 4.10 Training curves of BLEU-test scores on EMNLP WMT News dataset with different learning rates lr.

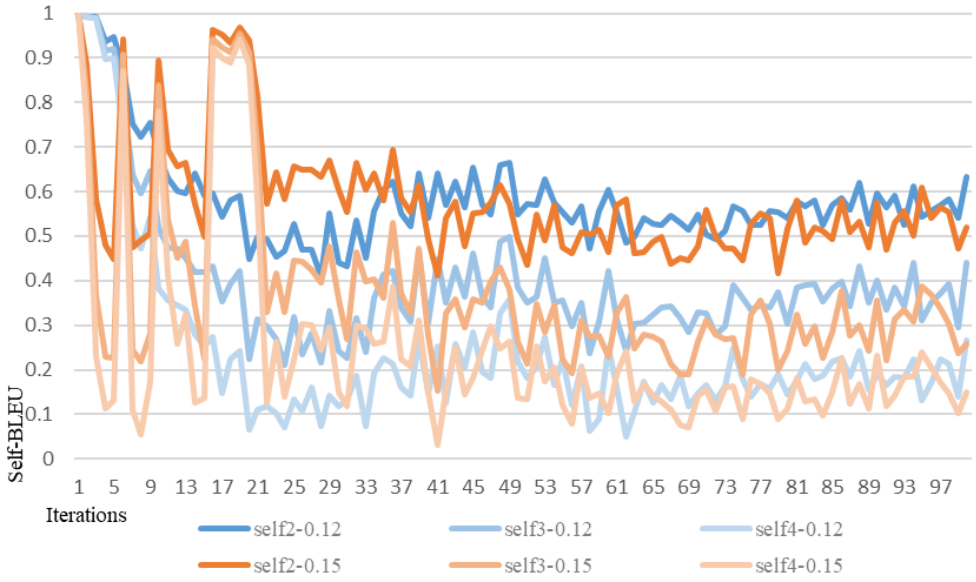

Figure 4.11 Training curves of Self-BLEU scores on EMNLP WMT News dataset with different learning rates lr.

We found that when the initial learning rate of the autoencoder is increased from 0.12 to 0.15, the performance of the generator will drop rapidly. At the same time, when lr is 0.15, the Self-BLEU curve no longer shows a gradual downward trend. Instead, it

becomes chaotic. These signs indicate that the generator cannot complete the generation task correctly, and the generated samples are chaotic. Based on the analysis, we believe there are two reasons for this. First, as the learning rate of the autoencoder is too high, the gap between the sentence vector of the real sample and the sentence vector generated by the generator is too large. There is little overlap between the distributions of the two sets of sentence vectors. The KL divergence is invalid in this case. The score given by the discriminator cannot guide the generator, and the generator cannot learn any meaningful information. Second, owing to the high learning rate of the autoencoder, the autoencoder falls into a saddle point. The autoencoder cannot effectively encode sentence vectors of real samples. The discriminator cannot provide helpful information to guide the generator. However, regardless of the problem, the generator can generate samples with higher scores when lr is 0.12 and the learning rate of the autoencoder used by the current model is 0.12.

4.8.3 Impact of the Initial Distribution

Data initialization also has a considerable impact on the performance of the model. Commonly used initialization distributions are as follows:

(1) zeros initialization: Initialize with zero matrix

(2) constant initialization: Initialize with specified constant

(3) ones initialization: Initialize with all-ones matrix

(4) identity initialization: Initialize with identity matrix

(5) random normal initialization: Initialize with random normal (Gaussian) distribution

(6) random uniform initialization: Initialize with uniform distribution in a given interval $[from, to]$

In this section, we discuss only the impact of Gaussian and Uniform initializations on the model. The scores of BLEU-test and Self-BLEU on the EMNLP WMT News dataset with different initialization distributions of noise input are shown in Figures 4.12 and 4.13. The orange curve (named $BLEUx-n$ and $selfx-n$) represents the scores of BLEU-test-2, 3, 4, 5 and Self-BLEU-2, 3, 4 when the model initialization

input follows a Gaussian distribution, and the blue curve (named $BLEUx-f$ and $selfx-f$) indicates that the model initialization input follows a Uniform distribution. The initialization parameters of Gaussian distribution are a $mean$ of 0 and a $standard\ deviation$ of 1, and the initialization parameters of Uniform distribution are a $from$ of 0 and a to of 1.

We found that the Gaussian distribution is slightly better regarding the quality of generated sentences but slightly worse in diversity than the Uniform distribution. Considering the comprehensive situation, we chose to use the Uniform distribution as the initial random noise distribution on the EMNLP WMT News dataset.

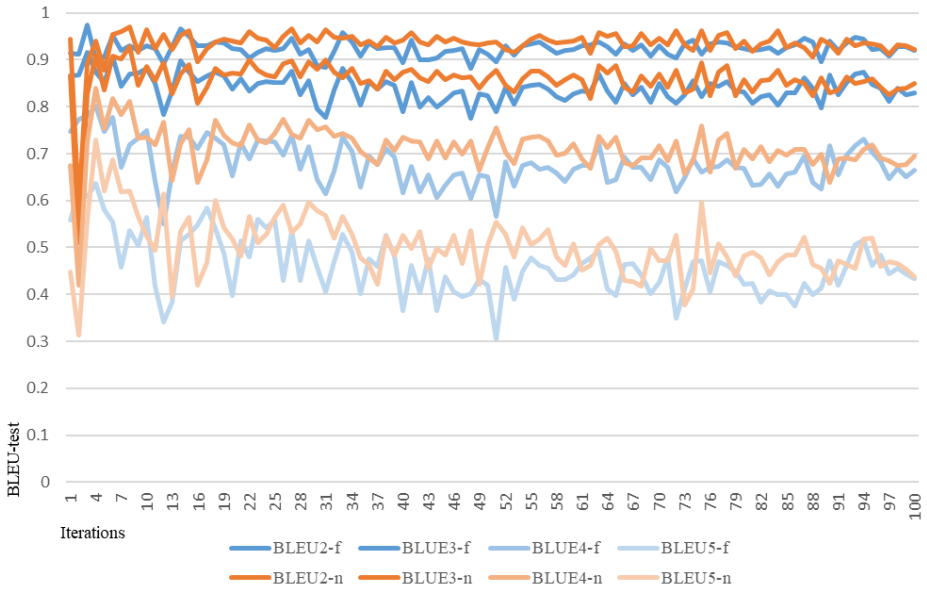

Figure 4.12 Training curves of BLEU-test scores on EMNLP WMT News dataset with different initialization distributions.

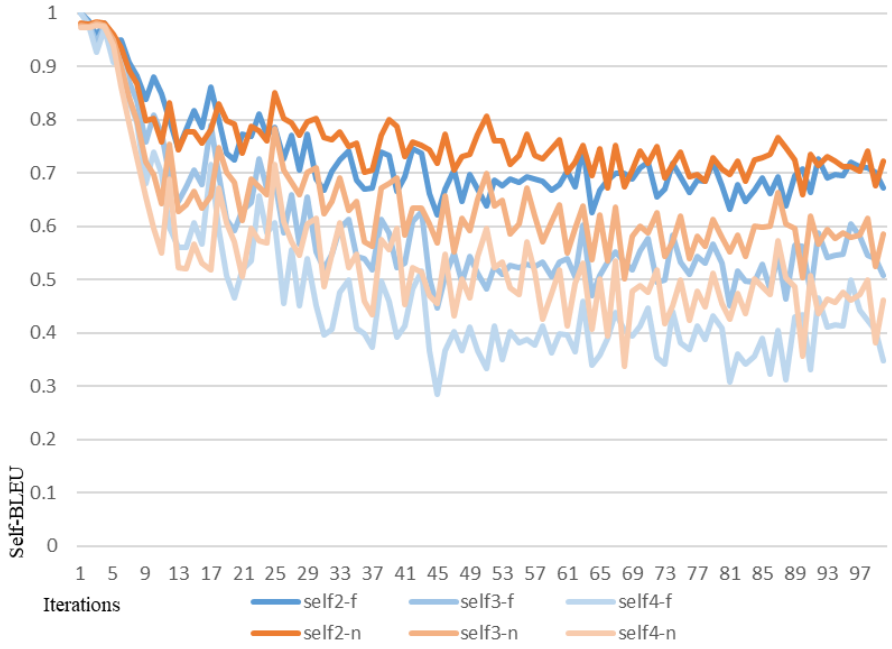

Figure 4.13 Training curves of Self-BLEU scores on EMNLP WMT News dataset with different initialization distributions.

4.9 Case Study

We randomly selected 10 generated samples from the short and long text datasets for the case study. The maximum lengths of generated sentences in the two datasets are 15 and 32, respectively. The performance of the model on the two datasets exhibited in good agreement with the experimental data.

4.9.1 The Generation Data from MSCOCO dataset

Table 4.6 lists the samples generated from the MSCOCO dataset. We set the Transformer autoencoder with 2 layers, 4 heads, and 512 hidden dimensions. The generator had 4 heads, a 256 head size, and 32 hidden dimensions. All the sentences were padded to their maximum length during training. From the table, we found that the generated sentences do not have obvious errors, such as identifiers “EOS” and “BOS.” In terms of sentence grammar, occasionally, the subject and object of the

sentence or the use of prepositions are inappropriate. The results show that the accuracy of the sentences generated by the model on the short sentence dataset was acceptable.

Meanwhile, we find that sentences starting with “a” accounted for the majority, which indicates that the diversity of sentences generated by the model on the short sentence dataset needs to be improved. We believe that there are two possible reasons for this finding. First, inappropriate parameter selection of the model makes the model too complex relative to the sample distribution of the MSCOCO dataset, resulting in overfitting. Second, we count 120,000 training data in the MSCOCO dataset and found that 81,450 sentences begin with “a,” and 4,950 sentences begin with “an.” The proportion of sentence types in the MSCOCO dataset is listed in Table 4.7. Therefore, the samples generated by the model will most likely start with “a.” In this case, we can only attempt to adjust the parameters to reduce the degree of overfitting of the model.

Table 4.6 The samples generated from the MSCOCO dataset.

Samples
a man and bathtub in a small wooden building
a man is standing on the side of a street
a tree on a grassy field next to some rocks
young man and woman in a small wooden building
a police performs a trick in makeshift lines window
a man is standing by a woman holding a child
a man is sitting on a bench with a dog
a close up of a person riding a bicycle
a man is standing in front of a bus
a group of people standing around a table eating food

Table 4.7 The percentage of sentence types in the MSCOCO dataset.

Beginning with	“a”	“an”	other
Percentage	67.875%	4.125%	28.00%

4.9.2 The Generation Data from EMNLP WMT News

Table 4.8 lists the samples generated from the EMNLP WMT News dataset during adversarial training. We set the Transformer autoencoder with 2 layers, 4 heads, and 512 hidden dimensions. The generator had 2 heads, a 256 head size, 32 hidden dimensions, and a batch size of 128. All the sentences will be padded to their maximum length during training. Our model performed better on the EMNLP WMTNews dataset than the MSCOCO dataset. From the table, we can see that the generated sentences do not have apparent errors. In terms of sentence grammar, occasionally, the subject and object of the sentence or the use of prepositions are inappropriate. Our model's performance benefits from the huge data volume of 270,000 and the excellent expressive ability of the generator in medium and long text generation.

Table 4.8 The samples generated from the EMNLP WMT News dataset.

Samples
It' s a big-time job to get a lot of people, but i' m not sure they're going to be a good team.
I' m not sure how much you can do with him, but i think he's going to get the ball.
We' re going to get a good job and get to the best, and we can do that.
the former secretary of state, said that the government would be able to take over the next few weeks, but it was not clear.
the first of the year, a new york businessman has been killed by a man who has been killed by a police officer in the past.
"we're going to get a lot of people in the world, and that's what we're doing," he said.
in the past, the number of people who have been killed in the past year has risen to more than a decade ago.
he said he would be able to get a job of keeping it in the car on the road, which is not the case.
" this is a very important part of the world, and it is a great opportunity to play," he said.

it is not clear whether the man was arrested on suspicion of murder, but it was not believed to be in the case.

4.10 Summary

This study proposes an improved model for text generation. We rebuilt the generator architecture with Multi-head Self-Attention to improve the text generation capabilities of the generator. Our model consists of a Transformer autoencoder, a generator with Multi-head Self-Attention, and a linear discriminator. We use the KL divergence as the GAN's loss functions. The encoder of the Transformer autoencoder is used to generate the distribution of real samples, and the decoder is used to decode the real samples encoding or the generated samples encoding into text. The loss function of the autoencoder is cross entropy. Our model has higher evaluation scores and diversity on MSCOCO and EMNLP WMT News datasets than existing models. Finally, we analyzed the influence of hyperparameters on the model. In future work, we will continue to conduct experiments on other datasets while seeking the best model parameters to obtain better performance.

Chapter 5

Conclusion and Future Work

With successive improvements, the theory behind GANs has gradually shown a high degree of potential, and many excellent variants of GANs have been developed for text generation. Nevertheless, many problems are yet to be addressed, including poor consistency, logical contradictions, insufficient information, and redundancy in generated content. All of these challenges indicate excellent development potential.

This thesis focuses on the GANs for unconditional text generation. This chapter summarizes the entire thesis and suggests future work.

5.1 Conclusion

This thesis focuses on the research of GANs for unconditional text generation. We proposed methods for generating high-quality and diverse texts. Overall, our work revolves around the remaining challenges in the field of unconditional text generation.

(1) Gradients cannot transfer appropriately between the generator and the discriminator.

(2) The training process of a GAN is unstable, and the generation task is significantly more complicated than discrimination, as the discriminator's guidance for the generator is too weak.

(3) Mode Collapse: The generator begins to degenerate, as it continuously generates the same samples, and cannot learn any meaningful information.

To address these challenges, we proposed two models.

(1) We proposed a novel architecture based on RelGAN and WGAN-GP, dubbed WRGAN, which effectively solves the issues identified above. We rebuilt the discriminator architecture with the 1-dimensional convolution of multiple kernel sizes and residual modules. Correspondingly, we modify the generator and discriminator loss functions with gradient penalty Wasserstein loss. Then, the discriminator and generator with relational memory were coordinated by Gumbel-softmax relaxation to train the GAN model on discrete data.

(2) We improved TILGAN for unconditional text generation by refactoring the generator. In short, we used Multi-head Self-Attention to replace the linear and BN layers to endow the generator with superior text generation capabilities. Our model consists of three components: a Transformer autoencoder, a Multi-head Self-Attention-based generator, and a linear discriminator. In the transformer autoencoder, the encoder generates the distribution of real samples, whereas the decoder decodes real or generated sentence vectors into text. The loss functions for autoencoder and GAN are cross entropy and KL divergence, respectively.

We then demonstrated the effectiveness of our proposed models by comparing their performance with that of existing models on multiple datasets. We also analyzed the effect of some typical hyperparameters in the proposed model. Our ablation experiments also demonstrated the effectiveness of the proposed generator network for unconditional text generation.

5.2 Future Work

This thesis comprises studies on unconditional text generation. There is still significant potential for improvement in this field, such as the creation of a model suitable for more datasets, particularly small datasets, and the development of a more suitable structure to enable the generation of high-quality samples. To apply the model to conditional text generation so that the model can generate text as we wish.

In the future, we will explore more effective methods for generating high-quality and diverse samples. However, there is still a significant gap between the language generated by machines, and that developed by humans. At the same time, we will neither limit the GAN models nor the field of NLG. Currently, the model based on multi-modal fusion is more in line with human cognition than a single-modal model. With the continuous development of technology, multi-modal generation will continue to be an important subject of research.

Bibliography

[1] E. Reiter, "20 Natural Language Generation," *The Handbook of Computational Linguistics and Natural Language Processing*, pp. 574, 2010.

[2] E. Reiter and R. Dale, "Building Applied Natural Language Generation Systems," *Natural Language Engineering*, vol. 3, no. 1, pp. 57-87, 1997.

[3] B. Zhang, D. Xiong, and J. Su, "Neural Machine Translation with Deep Attention," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 1, pp. 154-163, 2018.

[4] M. J. Nye, "Speaking in Tongues: Science's Centuries-Long Hunt for a Common Language," *Distillations*, vol. 2, no. 1, pp. 40-43, 2016.

[5] Y. Bengio, R. Ducharme, and P. Vincent, "A Neural Probabilistic Language Model," in *Proceedings of the Conference on Neural Information Processing Systems*, vol. 13, pp. 932-938, 2000.

[6] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," in *Proceedings of the International Conference on Learning Representations Workshop Track*, pp. 1-12, 2013.

[7] D. Elliott and F. Keller, "Image Description Using Visual Dependency Representations," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1292-1302, 2013.

[8] M. Hodosh, P. Young, and J. Hockenmaier, "Framing Image Description as a Ranking Task: Data, Models and Evaluation Metrics," *Journal of Artificial Intelligence Research*, vol. 47, pp. 853-899, 2013.

[9] A. Karpathy and L. Fei-Fei, "Deep Visual-Semantic Alignments for Generating Image Descriptions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3128-3137, 2015.

[10] R. Socher, A. Karpathy, Q. V. Le, C. D. Manning, and A. Y. Ng, "Grounded Compositional Semantics for Finding and Describing Images with Sentences,"

Transactions of the Association for Computational Linguistics, vol. 2, pp. 207-218, 2014.

[11] X. Chen and C. Lawrence Zitnick, "Mind's Eye: a Recurrent Visual Representation for Image Caption Generation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2422-2431, 2015.

[12] R. Barzilay and M. Lapata, "Aggregation via Set Partitioning for Natural Language Generation," in *Proceedings of the Human Language Technology Conference of the Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 359-366, 2006.

[13] J. Donahue *et al.*, "Long-Term Recurrent Convolutional Networks for Visual Recognition and Description," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2625-2634, 2015.

[14] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and Tell: a Neural Image Caption Generator," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3156-3164, 2015.

[15] C. Friedman, P. O. Alderson, J. H. Austin, J. J. Cimino, and S. B. Johnson, "A General Natural-Language Text Processor for Clinical Radiology," *Journal of the American Medical Informatics Association*, vol. 1, no. 2, pp. 161-174, 1994.

[16] H. Fang *et al.*, "From Captions to Visual Concepts and Back," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1473-1482, 2015.

[17] A. Moryossef, Y. Goldberg, and I. Dagan, "Step-by-Step: Separating Planning from Realization in Neural Data-to-Text Generation," in *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2267-2277, 2019.

[18] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning Representations by Back-Propagating Errors,” *Nature*, vol. 323, no. 6088, pp. 533-536, 1986.

[19] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735-1780, 1997.

[20] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to Sequence Learning with Neural Networks,” in *Proceedings of the Conference on Neural Information Processing Systems*, vol. 27, pp. 3104-3112, 2014.

[21] K. Cho *et al.*, “Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pp. 1724-1734, 2014.

[22] D. P. Kingma and M. Welling, “Auto-Encoding Variational Bayes,” in *Proceedings of the International Conference on Learning Representations*, pp.1-14, 2014.

[23] A. Vaswani *et al.*, “Attention is All You Need,” in *Proceedings of the Conference on Neural Information Processing Systems*, vol. 30, pp. 5998-6008, 2017.

[24] J. Sarzynska-Wawer *et al.*, “Detecting Formal Thought Disorder by Deep Contextualized Word Representations,” *Psychiatry Research*, vol. 304, pp. 114135, 2021.

[25] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, “Improving Language Understanding by Generative Pre-Training,” *URL https://s3-us-west-2.amazonaws.com/openai-assets/researchcovers/language-unsupervised/language_understanding_paper.pdf*, 2018.

[26] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, “Language Models are Unsupervised Multitask Learners,” *OpenAI blog*, vol. 1, no. 8, pp. 9, 2019.

[27] T. Brown *et al.*, “Language Models are Few-Shot Learners,” in *Proceedings of the Conference on Neural Information Processing Systems*, vol. 33, pp. 1877-1901, 2020.

[28] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding,” in *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4171-4186, 2019.

[29] R. McDonald, “Discriminative Sentence Compression with Soft Syntactic Evidence,” in *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 297-304, 2006.

[30] T. A. Cohn and M. Lapata, “Sentence Compression as Tree Transduction,” *Journal of Artificial Intelligence Research*, vol. 34, pp. 637-674, 2009.

[31] J. Clarke and M. Lapata, “Global Inference for Sentence Compression: an Integer Linear Programming Approach,” *Journal of Artificial Intelligence Research*, vol. 31, pp. 399-429, 2008.

[32] K. Thadani and K. McKeown, “Supervised Sentence Fusion with Single-Stage Inference,” in *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pp. 1410-1418, 2013.

[33] M. Elsner and D. Santhanam, “Learning to Fuse Disparate Sentences,” in *Proceedings of the Workshop on Monolingual Text-to-Text Generation*, pp. 54-63, 2011.

[34] C. Quirk, C. Brockett, and B. Dolan, “Monolingual Machine Translation for Paraphrase Generation,” in *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pp. 142-149, 2004.

[35] A. Fujita, K. Inui, and Y. Matsumoto, “Exploiting Lexical Conceptual Structure for Paraphrase Generation,” in *Proceedings of the International Conference on Natural Language Processing*, pp. 908-919, 2005.

[36] S. Narayan, S. B. Cohen, and M. Lapata, “Ranking Sentences for Extractive Summarization with Reinforcement Learning,” in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, vol. 1, pp. 1747-1759, 2018.

[37] L. Liu, Y. Lu, M. Yang, Q. Qu, J. Zhu, and H. Li, “Generative Adversarial Network for Abstractive Text Summarization,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, pp. 8109-8110, 2018.

[38] U. Hahn and I. Mani, “The Challenges of Automatic Summarization,” *Computer*, vol. 33, no. 11, pp. 29-36, 2000.

[39] J. Li, T. Tang, W. X. Zhao, and J. R. Wen, “Pretrained Language Models for Text Generation: a Survey,” in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pp. 4492-4499, 2021.

[40] L. Banarescu *et al.*, “Abstract Meaning Representation for Sembanking,” in *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pp. 178-186, 2013.

[41] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich, “A Review of Relational Machine Learning for Knowledge Graphs,” in *Proceedings of the IEEE*, vol. 104, no. 1, pp. 11-33, 2015.

[42] H. Mei, M. Bansal, and M. R. Walter, “What to Talk About and How? Selective Generation Using LSTMs with Coarse-to-Fine Alignment,” in *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 720-730, 2016.

[43] E. Reiter, “An Architecture for Data-to-Text Systems,” in *Proceedings of the 11th European Workshop on Natural Language Generation*, pp. 97-104, 2007.

[44] S. Antol *et al.*, “VQA: Visual Question Answering,” in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2425-2433, 2015.

[45] K. Xu *et al.*, “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention,” in *Proceedings of the International Conference on Machine Learning*, pp. 2048-2057, 2015.

[46] T. H. Huang *et al.*, “Visual Storytelling,” in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1233-1239, 2016.

[47] K. J. Shih, S. Singh, and D. Hoiem, “Where to Look: Focus Regions for Visual Question Answering,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4613-4621, 2016.

[48] Q. Wu, P. Wang, C. Shen, A. Dick, and A. Van Den Hengel, “Ask Me Anything: Free-Form Visual Question Answering Based on Knowledge from External Sources,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4622-4630, 2016.

[49] L. Yu, W. Zhang, J. Wang, and Y. Yu, “SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, pp. 2852-2858, 2017.

[50] I. J. Goodfellow *et al.*, “Generative Adversarial Nets,” in *Proceedings of the Conference on Neural Information Processing Systems*, pp. 2672-2680, 2014.

[51] T. Che *et al.*, “Maximum-Likelihood Augmented Discrete Generative Adversarial Networks,” *arXiv preprint arXiv:1702.07983*, 2017.

[52] R. Coulom, “Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search,” in *Proceedings of the International Conference on Computers and Games*, pp. 72-83, 2006.

[53] F. J. Xu, R. Dey, V. N. Boddeti, and M. Savvides, “RankGAN: a Maximum Margin Ranking GAN for Generating Faces,” in *Proceedings of the Asian Conference on Computer Vision*, pp. 3-18, 2018.

[54] J. Guo, S. Lu, H. Cai, W. Zhang, Y. Yu, and J. Wang, “Long Text Generation via Adversarial Training with Leaked Information,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, pp. 5141-5148, 2018.

[55] W. Fedus, I. Goodfellow, and A. M. Dai, “MaskGAN: Better Text Generation via Filling in the __,” in *Proceedings of the International Conference on Learning Representations*, pp, 1-17, 2018.

[56] J. Xu, X. Ren, J. Lin, and X. Sun, “Diversity-Promoting GAN: a Cross-Entropy Based Generative Adversarial Network for Diversified Text

Generation," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 3940-3949, 2018.

[57] K. Wang and X. Wan, "SentiGAN: Generating Sentimental Texts via Mixture Adversarial Networks," in *Proceedings of the International Joint Conferences on Artificial Intelligence*, pp. 4446-4452, 2018.

[58] W. Nie, N. Narodytska, and A. Patel, "RelGAN: Relational Generative Adversarial Networks for Text Generation," in *Proceedings of the International Conference on Learning Representations*, pp. 1-20, 2019.

[59] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved Training of Wasserstein GANs," in *Proceedings of the Conference on Neural Information Processing Systems*, vol. 30, pp. 5767-5777, 2017.

[60] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770-778, 2016.

[61] S. Diao, X. Shen, K. Shum, Y. Song, and T. Zhang, "TILGAN: Transformer-Based Implicit Latent GAN for Diverse and Coherent Text Generation," in *Proceedings of the Association for Computational Linguistics*, pp. 4844-4858, 2021.

[62] J. Li, Z. Tu, B. Yang, M. R. Lyu, and T. Zhang, "Multi-Head Attention with Disagreement Regularization," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2897-2903, 2018.

[63] T. Y. Lin *et al.*, "Microsoft COCO: Common Objects in Context," in *Proceedings of the European Conference on Computer Vision*, pp. 740-755, 2014.

[64] N. Rossenbach, J. Rosendahl, Y. Kim, M. Graça, A. Gokrani, and H. Ney, "The RWTH Aachen University Filtering System for the WMT 2018 Parallel Corpus Filtering Task," in *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pp. 946-954, 2018.

[65] K. Papineni, S. Roukos, T. Ward, and W. J. Zhu, "BLEU: a Method for Automatic Evaluation of Machine Translation," in *Proceedings of the 40th*

Annual Meeting of the Association for Computational Linguistics, pp. 311-318, 2002.

[66] A. Radford, L. Metz, and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," *in Proceedings of the International Conference on Learning Representations*, pp. 1-16, 2016.

[67] Y. Zhang, Y. Yin, R. Zimmermann, G. Wang, J. Varadarajan, and S. K. Ng, "An Enhanced GAN Model for Automatic Satellite-to-Map Image Conversion," *IEEE Access*, vol. 8, pp. 176704-176716, 2020.

[68] H. Emami, M. M. Aliabadi, M. Dong, and R. B. Chinnam, "SPA-GAN: Spatial Attention GAN for Image-to-Image Translation," *IEEE Transactions on Multimedia*, vol. 23, pp. 391-401, 2020.

[69] T. Karras, S. Laine, and T. Aila, "A Style-Based Generator Architecture for Generative Adversarial Networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4401-4410, 2019.

[70] X. Yu, Y. Qu, and M. Hong, "Underwater-GAN: Underwater Image Restoration via Conditional Generative Adversarial Network," in *Proceedings of the International Conference on Pattern Recognition*, pp. 66-75, 2018.

[71] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium," in *Proceedings of the Conference on Neural Information Processing Systems*, vol. 30, pp. 6629-6640, 2017.

[72] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative Adversarial Networks: an Overview," *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 53-65, 2018.

[73] M. Haidar, M. Rezagholizadeh, A. Do-Omri, and A. Rashid, "Latent Code and Text-Based Generative Adversarial Networks for Soft-Text Generation," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, vol. 1, pp. 2248-2258, 2019.

[74] C. de Masson d'Autume, S. Mohamed, M. Rosca, and J. Rae, "Training Language GANs from Scratch," in *Proceedings of the Conference on Neural Information Processing Systems*, vol. 32, pp. 4300-4311, 2019.

[75] A. Srivastava, L. Valkov, C. Russell, M. U. Gutmann, and C. Sutton, "VEEGAN: Reducing Mode Collapse in GANs Using Implicit Variational Learning," in *Proceedings of the Conference on Neural Information Processing Systems*, vol. 30, pp. 3010-3020, 2017.

[76] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio, "Generating Sentences from a Continuous Space," in *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pp. 10-21, 2016.

[77] C. J. Maddison, A. Mnih, and Y. W. Teh, "The Concrete Distribution: a Continuous Relaxation of Discrete Random Variables," in *Proceedings of the International Conference on Learning Representations*, pp. 1-20, 2017.

[78] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein Generative Adversarial Networks," in *Proceedings of the International Conference on Machine Learning*, pp. 214-223, 2017.

[79] X. Chen *et al.*, "Microsoft COCO Captions: Data Collection and Evaluation Server," *arXiv preprint arXiv:1504.00325*, 2015.

[80] X. Zhang and M. Lapata, "Chinese Poetry Generation with Recurrent Neural Networks," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pp. 670-680, 2014.

[81] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement Learning: a Survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237-285, 1996.

[82] V. Konda and J. Tsitsiklis, "Actor-Critic Algorithms," in *Proceedings of the Conference on Neural Information Processing Systems*, vol. 12, pp. 1008-1014, 1999.

[83] G. E. Hinton, A. Krizhevsky, and S. D. Wang, "Transforming Auto-Encoders," in *Proceedings of the International Conference on Artificial Neural Networks*, pp. 44-51, 2011.

[84] D. Donahue and A. Rumshisky, “Adversarial Text Generation without Reinforcement Learning,” *arXiv preprint arXiv:1810.06640*, 2018.

[85] A. S. Imran, R. Yang, Z. Kastrati, S. M. Daudpota, and S. Shaikh, “The Impact of Synthetic Text Generation for Sentiment Analysis Using GAN Based Models,” *Egyptian Informatics Journal*, pp. 1-11, 2022.

[86] D. Titone, T. Ditman, P. S. Holzman, H. Eichenbaum, and D. L. Levy, “Transitive Inference in Schizophrenia: Impairments in Relational Memory Organization,” *Schizophrenia Research*, vol. 68, no. 3, pp. 235-247, 2004.

[87] Y. Zhang, Y. Miyamori, S. Mikami, and T. Saito, “Vibration-Based Structural State Identification by a 1-Dimensional Convolutional Neural Network,” *Computer-Aided Civil and Infrastructure Engineering*, vol. 34, no. 9, pp. 822-839, 2019.

[88] F. Ren and Y. Zhou, “CGMVQA: a New Classification and Generative Model for Medical Visual Question Answering,” *IEEE Access*, vol. 8, pp. 50626-50636, 2020.

[89] D. P. Kingma and J. Ba, “Adam: a Method for Stochastic Optimization,” in *Proceedings of the International Conference on Learning Representations*, pp. 1-15, 2015.

[90] Y. LeCun, Y. Bengio, and G. Hinton, “Deep Learning,” *Nature*, vol. 521, no. 7553, pp. 436-444, 2015.

[91] O. Levy and Y. Goldberg, “Dependency-Based Word Embeddings,” in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, vol. 2, pp. 302-308, 2014.

[92] F. Ren and S. Xue, “Intention Detection Based on Siamese Neural Network with Triplet Loss,” *IEEE Access*, vol. 8, pp. 82242-82254, 2020.

[93] A. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, “Learning Word Vectors for Sentiment Analysis,” in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142-150, 2011.

[94] L. Theis, A. v. d. Oord, and M. Bethge, “A Note on the Evaluation of Generative Models,” in *Proceedings of the International Conference on Learning Representations*, pp. 1-10, 2015.

[95] Y. Zhu *et al.*, “Texygen: a Benchmarking Platform for Text Generation Models,” in *Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pp. 1097-1100, 2018.

[96] F. Ren and J. Deng, “Background Knowledge Based Multi-Stream Neural Network for Text Classification,” *Applied Sciences*, vol. 8, no. 12, pp. 2472, 2018.

[97] Z. Liu, J. Wang, and Z. Liang, “CatGAN: Category-Aware Generative Adversarial Networks with Hierarchical Evolutionary Learning for Category Text Generation,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 5, pp. 8425-8432, 2020.

[98] H. Y. Wu and Y. L. Chen, “Graph Sparsification with Generative Adversarial Network,” in *Proceedings of the 2020 IEEE International Conference on Data Mining*, pp. 1328-1333, 2020.

[99] L. Chen *et al.*, “Adversarial Text Generation via Feature-Mover’s Distance,” in *Proceedings of the Conference on Neural Information Processing Systems*, vol. 31, pp. 4666-4677, 2018.

[100] D. Liu and G. Liu, “A Transformer-Based Variational Autoencoder for Sentence Generation,” in *Proceedings of the 2019 International Joint Conference on Neural Networks*, pp. 1-7, 2019.

[101] Z. Wang, Z. Meng, K. Saho, K. Uemura, N. Nojiri, and L. Meng, “Deep Learning-Based Elderly Gender Classification Using Doppler Radar,” *Personal and Ubiquitous Computing*, vol. 26, no. 4, pp. 1067-1079, 2022.