Computers, Materials & Continua

Tech Science Press

Doi:10.32604/cmc.2025.063209

ARTICLE

GSPT-CVAE: A New Controlled Long Text Generation Method Based on T-CVAE

Tian Zhao*, Jun Tu*, Puzheng Quan and Ruisheng Xiong

School of Computer Science, Hubei University of Technology, Wuhan, 430070, China

*Corresponding Authors: Tian Zhao. Email: 102201084@hbut.edu.cn; Jun Tu. Email: tujun@mail.hbut.edu.cn

Received: 08 January 2025; Accepted: 11 April 2025; Published: 09 June 2025

ABSTRACT: Aiming at the problems of incomplete characterization of text relations, poor guidance of potential representations, and low quality of model generation in the field of controllable long text generation, this paper proposes a new GSPT-CVAE model (Graph Structured Processing, Single Vector, and Potential Attention Computing Transformer-Based Conditioned Variational Autoencoder model). The model obtains a more comprehensive representation of textual relations by graph-structured processing of the input text, and at the same time obtains a single vector representation by weighted merging of the vector sequences after graph-structured processing to get an effective potential representation. In the process of potential representation guiding text generation, the model adopts a combination of traditional embedding and potential attention calculation to give full play to the guiding role of potential representation for generating text, to improve the controllability and effectiveness of text generation. The experimental results show that the model has excellent representation learning ability and can learn rich and useful textual relationship representations. The model also achieves satisfactory results in the effectiveness and controllability of text generation and can generate long texts that match the given constraints. The ROUGE-1 F1 score of this model is 0.243, the ROUGE-2 F1 score is 0.041, the ROUGE-L F1 score is 0.22, and the PPL-Word score is 34.303, which gives the GSPT-CVAE model a certain advantage over the baseline model. Meanwhile, this paper compares this model with the state-of-the-art generative models T5, GPT-4, Llama2, and so on, and the experimental results show that the GSPT-CVAE model has a certain competitiveness.

KEYWORDS: Controllable text generation; textual graph structuring; text relationships; potential characterization

1 Introduction

In today's information age, Deep Learning (DL), as one of the important branches of machine learning, has made remarkable achievements in the fields of image recognition, speech processing, and natural language processing (NLP). Meanwhile, text generation, as one of the key issues in natural language processing, focuses on how to make computers generate natural, coherent, and meaningful texts. With the continuous progress of natural language processing technology, text generation models have achieved remarkable success in dialog systems, automatic writing, and translation. However, in practical applications, users increasingly need to control the generated text, as traditional text generation models often lack sufficient controllability regarding generated content, style, and tone. In this context, controlled text generation has become a highly anticipated research direction, aiming to enhance the controllability of model-generated text and better adapt to the specific needs of users.

Controlled text generation (CTG) task [1] refers to the generation of natural language text that satisfies constraints such as topic, emotion, and style while ensuring grammatical correctness. It is an essential tool in human-computer interaction [2]. Early approaches were mainly based on templates, which could only control the structure of the text. Thus, the generated text lacked diversity in expression. In recent years, with the complexity of application scenarios, the demand for controllability of various aspects of text has been increasing. To meet these flexible and diverse needs, data-driven neural network methods have become the basic methods for controllable text generation. Large-scale pre-trained language models can generate fluent text that meets constraints through different strategies [3] such as fine-tuning [4] and cue learning.

Control conditions in CTG can be categorized as explicit or implicit. Explicit control refers to providing well-defined instructions through human-computer interaction, such as input prompts, to guide the model in generating text with a specific style, like a Shakespearean or humorous tone [5]. In contrast, implicit control ensures that the generated content adheres to certain standards even when these requirements are not explicitly stated. This includes maintaining non-toxic, inoffensive, and nondiscriminatory language. For example, intelligent customer service systems should consistently convey a positive and optimistic tone to improve user experience. To meet these implicit expectations, the model needs to autonomously adjust its output and prevent content that may cause social concerns.

Research on controlled text generation broadly encompasses three foci of interest from different dimensions. The first focus of attention is primarily on the relationship between neighboring texts (short text environment), such as story completion, poetry generation, etc. The main methods used are fine-tuning based on pre-trained models [6,7], cue learning [8,9,10], conditional pre-trained language models, attribute bootstrapping [11–14], and adversarial generative structures [15,16]; The second focus is on textual diversity control, mainly using approaches based on sampling strategies [17,18], variational autoencoders [19,20], and diffusion models [21,22]; The third focus of attention is then on the global relations of the text (extended textual contexts), and the mainstream methods include multi-step generation strategies [23–26], semantic consistency discriminators [27], and relevance-based knowledge enhancement [28–30].

2 Related Work

With the rapid advancement of Large Language Models (LLMs) and their expanding role in NLP, significant improvements have been made in text generation quality. However, real-world applications impose more complex and rigorous content requirements. For instance, in finance [31] and news reporting [32], models must not only prevent misleading or biased outputs but also align precisely with specific conditions and user preferences. These may involve mimicking a particular writing style or incorporating poetic elements. To address such needs, Controllable Text Generation also known as Controlled or Constrained Text Generation has emerged, ensuring generated content meets both quality standards and application specific demands.

However, LLMs excel in logical reasoning, text analysis, and problem-solving [33] but struggle with Controlled Sentiment and Style Generation (CTG) tasks, which demand accuracy and precise sentiment-tone alignment. A key limitation is their insufficient grasp of subtle emotional expressions; while capable of simulating emotions, they often misjudge intensity and shifts, leading to text that may be too flat or too strong, undermining emotional precision in contexts like customer service or literary creation [34]. Additionally, LLMs struggle with style control, as their mimicry relies on statistical patterns rather than proper stylistic understanding. This makes them ineffective in blending complex styles, such as professionalism with humor, often producing content that fails to meet expectations. Furthermore, CTG tasks require high context sensitivity, yet LLMs usually falter in maintaining emotional coherence in long texts due to context misinterpretation or memory limitations. The subjective nature of CTG evaluation further complicates optimization, as user preferences vary across cultural and individual factors, and LLMs lack the adaptability to these differences. Thus, CTG expands LLM applications and exposes deficiencies in emotion, style, and context handling. Enhancing LLMs for CTG requires larger and higher-quality datasets and advanced sentiment modeling and context management to meet its intricate demands [35].

Therefore, most researchers nowadays will use a combination of pre-trained LLM with latent variable models, the most typical of which are VAEs [36]. However, the use of pre-trained LLM in combination with latent variable models leads to the following two main problems: a) The encoders of large pre-trained models such as Transformer process the text mainly in a linearized way such as linear transformations [37]. In contrast, their encoders usually use the last hidden state to generate the latent space, producing a vector sequence [38]. The result of this is that a more comprehensive representation of the relationships between the input texts is not available, which leads to less accurate latent representations obtained in the subsequent hidden space; b) The latent representations received from the latent variable model fail to achieve the expected results in terms of validity and controllability in the subsequently guided text generation after adding the decoding process to the pre-trained model [39].

At the same time, controlled text generation faces key challenges affecting practical applications. In automated content creation (news, blogs, novels), incomplete text representation may omit key facts, reducing credibility. At the same time, poor potential guidance can cause deviations from user intent, leading to unfocused blog topics. Low generation quality manifests as repetitive or unclear text, affecting user acceptance. In education, incomplete representation may omit key concepts, poor guidance may result in mismatched difficulty levels, and low quality can introduce grammar errors, reducing readability. In customer service, incomplete representation leads to inaccurate responses, poor guidance prevents adaptation to user needs, and low quality can make responses unprofessional. In healthcare, missing contextual details in generated medical advice can mislead patients, poor guidance may cause content to deviate from the intended topic, and low quality may result in incorrect medical terminology, reducing trust. These issues persist in traditional generative models, as shown in Table 1.

Table 1: Generation failures of traditional generative models

Generted text	**Error analysis**
Quantum entanglement is a new technology that enables instantaneous communication over infinite distances, allowing information to be transmitted faster than the speed of light. Many modern devices, such as smartphones and Wi-Fi routers, already integrate quantum entanglement for ultra-secure encryption.	The generated text contains physically incorrect content, i.e., quantum entanglement cannot be used for FTL communication and is not widely used in smart devices. The model, in the absence of explicit knowledge constraints, may generate content that appears reasonable but is actually wrong.
The artificial intelligence has improve diagnostic accuracy by analyzing large amounts of datas, it also assisting in surgeries and patient monitoring. The deep learning algorithms are very helps in early disease detection.	The text contains grammatical errors (e.g., 'datas' should be 'data', 'is very helpful' should be 'is very helpful'), which may be due to the model's failure to learn the grammar rules correctly, or instability in the decoding phase.

(Continued)

Table 1 (continued)

Generted text	Error analysis
Renewable energy has significantly reduced electricity costs in developing countries. However, due to its high initial costs, it has led to increased electricity prices for consumers. This economic burden makes renewable energy less accessible to rural populations, yet it remains the cheapest option available.	The text contains contradictory logic, stating on the one hand that renewable energy reduces the cost of electricity, but on the other hand stating that the high initial cost leads to an increase in the cost of electricity, and then ultimately stating that it is the 'cheapest option'. The model may be locally consistent in the generation of long texts, but the overall semantics are conflicting.

Given the current problems of controllable long text generation, the main content of the research in this paper is the effectiveness and controllability of generating long text. The research method is to abandon the traditional encoder structure and combine the pre-training model with the text graph-structured latent variable model by graph-structured processing of the input text as well as modeling the graph-structured text relations, which can not only utilize the pre-trained language model as a reliable tool for feature extraction or a reliable tool for text decoding, it can also use the hidden space of the latent variable model to get the comprehensive features of the text, which helps to improve the controllability of long texts.

In this paper, the GPT-2 model is chosen as the decoder for the GSPT-CVAE model because 1) GPT-2 is an autoregressive-based language model that produces text by continuously generating sequences of words [40]. In contrast, models such as BERT are based on Masked Language Modeling (MLM), which can only create parts of text. As a result, GPT-2 is more coherent and natural in generating text and can produce more diverse text. 2) Compared to traditional VAE, GPT-2 adopts a parameter-sharing strategy. This means that the parameters of the model are used in both parts of the encoder and decoder, which reduces the number of parameters that need to be trained and improves the efficiency and generalization of the model. 3) GPT-2 is a pre-trained language model that can be used directly for generating text without additional pre-training or fine-tuning. In contrast, models such as BERT usually require pre-training or fine-tuning on specific tasks, which requires more time and computational resources. Therefore, GPT-2 not only meets the requirements of GSPT-CVAE continuous text generation but also achieves a good balance between resource utilization and performance and is thus selected as the decoder for this model.

2.1 Conditional Variational Auto Encoder

In this paper, we choose controlled long text generation based on the prompt method, i.e., generating open-domain long text based on the input prompt text, and the hidden space of the latent variable model C-VAE is utilized to obtain latent representations from textual relations. Fig. 1 illustrates the model diagram of the variational autoencoder VAE, whose architecture consists of an encoder (inference network) and a decoder (generation network), where the encoder maps the input data x to the probability distribution $q_\varphi(z|x, y)$ of the latent variable z, and then samples from this distribution are obtained, and the latent variable z is obtained through the computation of the mean, variance, and parameters. The decoder generates the reconstructed probability distribution $p_\theta\left(x' \mid y, z\right)$ of the data x', using z as input. The traditional VAE decoder $p_\theta(x|y, z)$ is usually in autoregressive form; in this paper, we use a pre-trained GPT-2-based model as the decoder.

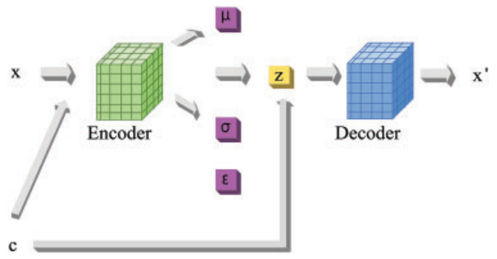

Figure 1: Diagram of the basic structure of the C-VAE model used in our GSPT-CVAE model. In controlled text generation, x and x' refer to prompt text and generated text, respectively. c is the condition, μ is the mean value, σ is the variance, z is the latent variable

The goal of C-VAE training is to maximize the data log-likelihood $E_{x,y\sim D}\left[\log p_{\theta}\left(x\right)\right]$. However, since posterior inference is difficult to achieve, C-VAE approximates the posterior distribution through variational inference methods to maximize the lower bound on the marginal probability of the observed data (ELBO):

$$\mathbb{E}_{\boldsymbol{x},\boldsymbol{y}\sim\mathscr{D}}\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{y}) \geq \mathbb{E}_{\boldsymbol{x},\boldsymbol{y}\sim\mathscr{D}}\left[\mathbb{E}_{\boldsymbol{z}\sim q_{\phi}(z|\boldsymbol{x},\boldsymbol{y})}\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{y},\boldsymbol{z})\right] - \mathbb{E}_{\boldsymbol{x},\boldsymbol{y}\sim\mathscr{D}}\left[\mathrm{KL}\left(q_{\phi}(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{y}) \parallel p(\boldsymbol{z}|\boldsymbol{y})\right)\right] \tag{1}$$

Here, $\mathbb{E}_{\boldsymbol{x},\boldsymbol{y}\sim\mathscr{D}}\left[\mathrm{KL}\left(q_{\phi}(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{y}) \parallel p(\boldsymbol{z}|\boldsymbol{y})\right)\right]$ is the KL dispersion between the latent space distribution $q_{\phi}(z|x)$ and the prior distribution $p(z)$ learned by the encoder, which is used to ensure the continuity and compactness of the latent space. x is the input text, and y is the condition.

2.2 Graph Convolutional Networks (GCN) Encoder

Graph Convolutional Encoder is a model for learning graph data representation by combining a node's neighbor information and features to generate a low-dimensional representation of the node. The focus of this paper is on the global relations of the text. Fig. 2 illustrates a graph convolutional encoder with a two-layer convolutional network. Each word in the input text is treated as a node in the graph, the node features are vector representations of the input text, and the connecting lines between the nodes are the values of the elements in the resulting adjacency matrix A. After the text passes through the graph convolutional network, the text features are updated to obtain a new vector representation of the text.

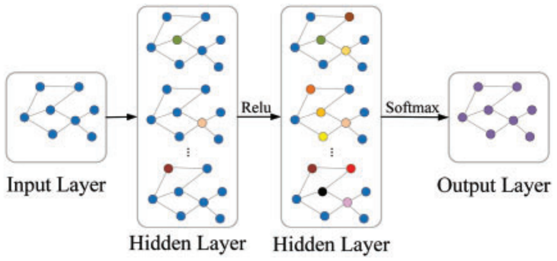

Figure 2: Graph convolutional encoder (with two layers of convolutional networks)

The GCN formula is as follows:

$$H^{l+1} = \sigma(\tilde{D}^{\frac{1}{2}}\tilde{A}\tilde{D}^{\frac{1}{2}}H^{l}W^{l}) \tag{2}$$

Here $\tilde{A}$ is the new adjacency matrix obtained by adding the unit matrix to the adjacency matrix A, D is the degree matrix of the adjacency matrix A, W is the weight parameter matrix of the Lth layer, σ is a nonlinear activation function, e.g., ReLU(Rectified Linear Unit). For a two-layer convolutional network, the forward propagation formula is:

$$z = f(X, A) = softmax(\tilde{A}ReLU(\tilde{A}XW^0)W^1) \tag{3}$$

For the loss function, the traditional cross-entropy function is chosen.

The rest of the paper is organized as follows. Section 3 details the overall architectural design of the model proposed in this paper. Section 4 is the experimental part and comparative analysis of the experimental results. Section 5 summarizes the work of this paper and provides an outlook on the next steps.

3 Model Architecture (GSPT-CVAE)

The overall model diagram of the GSPT-CVAE model proposed in this paper is shown in Fig. 3. The model consists of three main modules:

- In the encoding stage, the text is graph-structured to obtain the graph-structured representation of the text, then passes through the GCN encoder to collect text node representations, i.e., the vector sequences of the text representations, and proceeds with the subsequent work.
- The vector sequences are combined into a single vector representation using the vector merging layer, and this representation is then fed into the linear layer to predict both the prior and posterior distributions, which results in the latent representation Z.
- In the decoding stage, The latent representation Z is added to the Transformer decoder through a combination of embedding and latent attention computation to guide subsequent decoding generation.

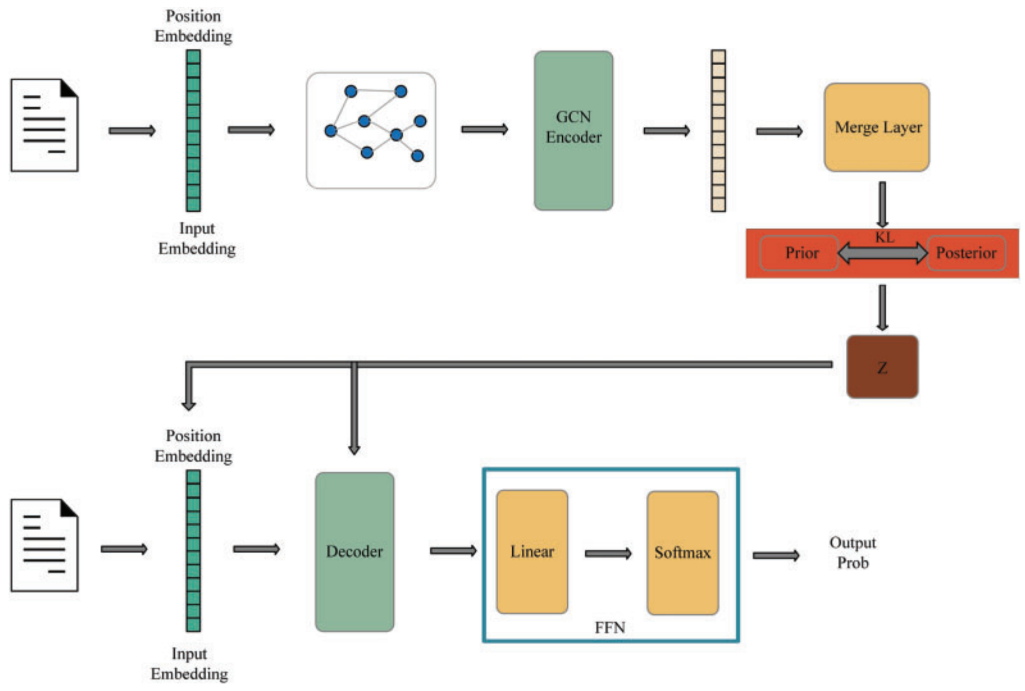

Figure 3: Graph convolutional encoder (with two layers of convolutional networks)

3.1 Textual Graph Structuring

After the input text data is encoded, the resulting text vector can accurately represent the nodes in the graph, i.e., the feature matrix X. Thus, only the adjacency matrix A of the text nodes needs to be constructed. Afterward, the feature matrix X and the adjacency matrix A are inputted into the GCN encoder with a two-layer convolutional network to obtain new node representations containing comprehensive relationships among the texts.

Commonly used methods for structuring text graphs include dependent syntactic analysis trees, sentence component analysis, AMR (Abstract Meaning Representation) graphs, information extraction graphs, knowledge graphs, etc. Still, all of these methods have some limitations. For example, sentence component analysis may have difficulty in describing sentence components in specific languages and text types [41]; the construction of AMR graphs relies on the subjective interpretation of manual annotators, which may lead to inconsistency and subjectivity problems [42]; information extraction graphs usually focus on the extraction of entities and relations, which may ignore important information in the context, etc. [43].

A filter exists in a convolutional neural network, which extracts information from an image through a constantly moving fixed-size filter window. Inspired by the filter window, this paper proposes to design a "sliding window" to traverse the input text. In this window, the adjacency matrix A is constructed by the features of the text, and the word embedding of the text is used as the feature matrix X of the text, thus completing the graph structuring of the text.

However, the sliding window approach also introduces a new problem: the relationships between texts are far or near. When using a sliding window to obtain relationships, if the window size is too small, the complete relationship information cannot be obtained; if the window size is too large, unnecessary information, i.e., noise, is introduced. This affects the construction of the final graph.

In response to the new problems triggered, a dynamic sliding window can be designed to solve the problem. The so-called dynamic sliding is based on the strength of the textual relationship, which adjusts the window size in real time to capture the textual relationship under different relationship strengths. The specific process is as follows: set the initial window size k (e.g., the initial size is set to 3), and then traverse the entire text sequence, the three words within the first window, the average cosine similarity value can be obtained after the cosine similarity is calculated using its vector representation. Set a threshold T when the average cosine similarity is greater than the threshold T, increase the window size, and vice versa, reduce the window (minimum of 2) until traversing the entire text sequence, and ultimately get the adjacency matrix A, in which each element of the element matrix in the adjacency matrix A represents the strength of the relationship between word i and word j, the larger the value indicates that the more similar the two words are to each other, and the smaller the value indicates that the more the similarity is lower. The specific process is shown in Fig. 4.

Traditional Transformer only relies on Positional Encoding and Self-Attention to model inter-word relationships, which makes it easy to ignore long-distance dependencies. In this paper, the method adopts GCN to model the text as a Graph Representation. It dynamically connects the related words through the sliding window to enhance the global dependency modeling. In the dynamic sliding window, the strengths and weaknesses of the relationships between the texts are further recorded, and the final graph structure representation is improved according to the strengths and weaknesses of the relationships so that the resulting textual relationships have better strengths and weaknesses compared to the textual relationship representations obtained from traditional coding. The following demonstrates how the dynamic sliding window captures strong and weak textual relations.

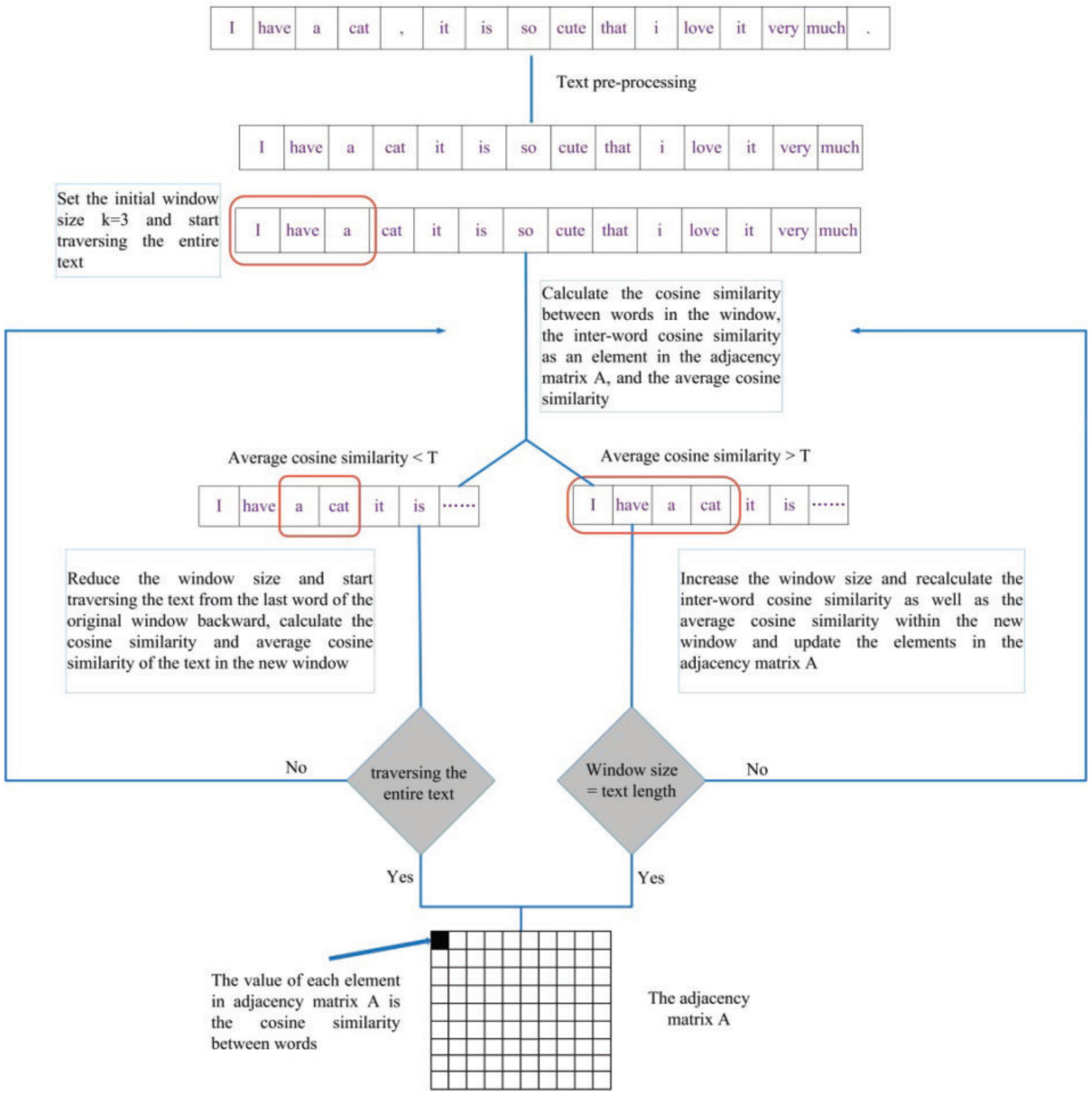
I have a cat , it is so cute that i love it very much .
Text pre-processing
I have a cat it is so cute that i love it very much
Set the initial window size k=3 and start traversing the entire text
Calculate the cosine similarity between words in the window, the inter-word cosine similarity as an element in the adjacency matrix A, and the average cosine similarity
Average cosine similarity < T
Average cosine similarity > T
Reduce the window size and start traversing the text from the last word of the original window backward, calculate the cosine similarity and average cosine similarity of the text in the new window
Increase the window size and recalculate the inter-word cosine similarity as well as the average cosine similarity within the new window and update the elements in the adjacency matrix A
traversing the entire text
Window size = text length
No
No
Yes
Yes
The value of each element in adjacency matrix A is the cosine similarity between words
The adjacency matrix A

Figure 4: Dynamic sliding window to construct the adjacency matrix

Now consider the set of word vectors for two windows W_1 and W_2, $\mathrm{W}_2 > \mathrm{W}_1$ and containing W_2 contains W_1, and the average cosine similarity between them is:

$$AvgCosineSimilarity(W_1, W_2) = \frac{1}{\min(|W_1|, |W_2|)} \sum_{a \in W_1} \sum_{b \in W_2} \text{Cosine Similarity}(a, b) \tag{4}$$

Here, W_1, W_2 are the window sizes, respectively. Define a threshold T when *AvgCosineSimilarity* $(W_1, W_2) > T$, it represents a higher similarity between the two windows, which means that the textual contexts within the two windows are similar, i.e., better textual relations can be obtained when W_1 is increased to the size of W_2. Similarly, when *AvgCosineSimilarity* $(W_1, W_2) < T$, which represents a lower similarity between the two windows, suggests that there is a large gap between the textual contexts within the two windows, at which point the window can be narrowed down to focus on a tighter context. This proves that this paper's dynamic sliding window approach can better capture the semantic relationships between texts.

3.2 Vector Sequence Merge

When data is encoded, traditional Transformer models use only the last hidden state from the encoder to generate the latent space. After the final self-attention layer, a sequence of vectors is obtained with the number of vectors as the number of input tokens. However, the vector sequence obtained after such encoding may not be sufficient to summarize continuous data and retain long-term knowledge, mainly due to the limitations of the self-attention mechanism, positional encoding, etc. This paper proposes a vector sequence merging method to merge vector sequences into a single vector through attention scores for subsequent potential representation generation work. A single vector contains both global and local input text information, and the single vector can better process and retain feature information than vector sequences. Also, due to the use of Attention Score Weighted Merging, each feature is weighted and merged during vector sequence merging, thus further emphasizing important features and weakening minor features. The following is a validation of the methodology of this paper in terms of both weights and expectations.

First, from the point of view of weights, suppose there is an input sequence $X = \{x_1, x_2, x_3 \ldots x_n\}$, where x denotes the xth element in the sequence. Using the self-attention mechanism, it is possible to compute a vector A_i of attention weights at each position and then apply these weights to the original vector sequence. This results in a weighted vector and Z, where the weighted sum for each position i is computed as follows:

$$Z_i = \sum_{j=1}^{n} A_{i,j} \cdot x_j \tag{5}$$

Here, $A_{i,j}$ is the attentional weight from position i to position j. To combine this sequence into a single vector, consider weighting and summing all the positions, i.e., $final\_output = \sum_{i=1}^{n} z_i$ rearrange this equation and introduce attentional weights:

$$final\_output = \sum_{i=1}^{n} \left(\sum_{j=1}^{n} A_{i,j} \cdot x_j \right) \tag{6}$$

Obtained by swapping the summation order:

$$final\_output = \sum_{j=1}^{n} \left(\sum_{i=1}^{n} A_{i,j} \right) \cdot x_j \tag{7}$$

The portion in parentheses is the sum of all the attentional weights for position j, which can be thought of as the composite weight for position j. The weight of position j is the sum of all the attentional weights for position j. Let $C_j = \sum_{i=1}^{n} A_{i,j}$ The above equation can be simplified as:

$$final\_output = \sum_{j=1}^{n} C_j \cdot x_j \tag{8}$$

This equation shows that the weighted composite C_j obtained through the self-attention mechanism and the weighted sum of the original vector sequence X form the final representation. These synthesized weights can better capture the relationships between different positions in the sequence, thus providing richer semantic information.

Moving on to the expectation perspective, suppose again that there is an input sequence X, denoted $\{x_1, x_2, \ldots, x_T\}$, of length T. Each x_t is a vector. Now compute the attentional weight α_t, where the attentional weight α_t is denoted:

$$\alpha_t = \frac{\exp(e_t)}{\sum_{k=1}^{T} \exp(e_k)} \tag{9}$$

Here, e_t is some rating or score associated with x_t can be computed using any appropriate mechanism (e.g., dot product, scaled dot product, etc.). The attention-weighted average vector is then computed:

$$AttentionAvg = \sum_{t=1}^{T} \alpha_t * x_t \tag{10}$$

First, the attention weights are of the nature: $\sum_{t=1}^{T} \alpha_t = 1$, this is because the weights are standardized probability distributions. Next, the expected value of the weighted average vector of attention is computed:

$$\begin{aligned}\mathbb{E}[AttentionAvg] &= \mathbb{E}\left[\sum_{t=1}^{T} \alpha_t \cdot x_t\right] \\ &= \sum_{t=1}^{T} \mathbb{E}\left[\alpha_t \cdot x_t\right] \\ &= \sum_{t=1}^{T} \mathbb{E}[\alpha_t] \cdot \mathbb{E}[x_t] \quad \text{(linear property)} \\ &= \sum_{t=1}^{T} \mathbb{E}\left[\alpha_t\right] \cdot x_t \quad \text{(Definition of expectation)} \\ &= \sum_{t=1}^{T} \left(\frac{\exp(e_t)}{\sum_{k=1}^{T} \exp(e_k)}\right) \cdot x_t\end{aligned} \tag{11}$$

By normalizing α_t, the weighted average vector of expected attention will be more concentrated on the parts with higher scores e_t, i.e., the model will pay more attention to the tasks' relevant parts.

3.3 Embedding Combined with Latent Attention Computation

After obtaining the potential representation Z of the potential space, Z is usually added to the subsequent decoding and generation process by either concatenating Z with the input vectors of the decoder or multiplication operations, etc. or by fusing Z with each layer of the decoder according to the attention weights. In this paper, we not only use the method of concatenating Z with the input vectors but also add Z to the K and V matrices of the attention computation to obtain new K' and V' matrices and compute the attention. In this way, Z is fully integrated into the decoder generation process for better control of text generation.

For the traditional embedding approach, the drawback is the introduction of noise in the self-attention computation. The proof is as follows: e_i, e_j is defined as the marker embedding of the two inputs, and $\alpha_{i,j}$ is the attention weight of the ith and jth markers. According to the self-attention formula.

$$Attention(Q, K, V) = soft\max\left(\frac{Q^T K}{\sqrt{d}}\right) V \tag{12}$$

The expression $\alpha_{i,j}$ can be obtained as:

$$\alpha_{i,j} = \left(W^q e_i\right)^T \left(W^k e_j\right) = e_i^T \left(W^q\right)^T W^k e_j \tag{13}$$

W^q and W^k are the parameter matrices used to map the input marker embedding into the query (Q) and key (K) spaces. Denoting the right-hand side of this equation as (e_i, e_j), Z is embedded directly into the tags according to the embedding, i.e.:

$$\alpha_{i,j} = [W^q(e_i + z)]^T[W^k(e_j + z)] = (e_i, e_j) + (e_i, z) + (z, e_j) + (z, z) \tag{14}$$

It can be noted that the resulting equation introduces a redundant term (z, z) introduces additional noise to the attention mechanism.

To solve the problems caused by the embedding method, this paper introduces potential attention computation while using the embedding method; specifically, the potential representations obtained previously are divided into L vectors $[Z_1, Z_2 \dots Z_L]$, and the potential representations are merged into the original self-attention computation by adding the potential representations to the matrices K and V in each layer of the attention computation.

Integrate Z into K and V to get a new K′ and V′

$$K' = \begin{pmatrix} Z_K \\ K \end{pmatrix} \in \mathbb{R}^{(1+l)\times d} \quad V' = \begin{pmatrix} Z_V \\ V \end{pmatrix} \in \mathbb{R}^{(1+l)\times d} \tag{15}$$

The attention score α can be denoted as $\alpha = (Q * K')$, this is the first time Z is fused with the feature vector, after which the attention score is multiplied by the new value matrix in the subsequent self-attention computation. $Attention = (\alpha * V')$, this is the second time Z is fused with the feature vector, and in this computation, the value matrix V' already contains information about the latent variable Z. When the embedding method is used again, the effect of noise is resolved.

$$\begin{aligned} &[(e_i, e_j) + (e_i, z) + (z, e_j) + (z, z)] * (V') \\ &= [(e_i, e_j) + (e_i, z) + (z, e_j) + (z, z)] * \begin{pmatrix} Z_V \\ V \end{pmatrix} \\ &= (e_i, e_j) * \begin{pmatrix} Z_V \\ V \end{pmatrix} + (e_i, z) * \begin{pmatrix} Z_V \\ V \end{pmatrix} + (z, e_j) * \begin{pmatrix} Z_V \\ V \end{pmatrix} + (z, z) * \begin{pmatrix} Z_V \\ V \end{pmatrix} \end{aligned} \tag{16}$$

It can be seen that in the results of the above equation, the noise (z, z) present in the original embedding method is finally multiplied by a matrix that contains information about the incorporated latent variable Z. Therefore, this item is equivalent to the weighting of the latent variable Z and is no longer noisy.

4 Results and Discussion

4.1 Dataset Selection

The datasets used in this paper are Arxiv [44], Yelp [45], WritingPrompts [46], and WikiPlots [47]. In this paper, the Arxiv dataset is used to generate controlled text, and the Arxiv dataset is divided into a training dataset, testing dataset, and validation dataset in the ratio of 80:10:10. Arxiv is an online dataset that extracts abstracts from the 'Arxiv' articles. Arxiv is an online dataset for extracting abstracts from 'arxiv' articles. Arxiv mainly searches for topic queries in arxiv and then collects article content-matching Abstracts. This paper selects article types with keywords 'artificial intelligence,' 'computer vision,' and 'text generation.' Meanwhile, this paper experiments with the GSPT-CVAE model on WritingPrompts and WikiPlots datasets to show the model's performance under different data domains. On the other hand, the Yelp dataset is mainly used for pre-experimental analysis and textual relationship validation visualization experiments.

4.2 Selection of Comparison Methods

The comparison methods chosen for this paper are as follows:

- FIST: This method is a technique for fine-tuning natural language processing models by introducing special tokens to guide the model to focus on specific tasks or samples during the fine-tuning process [48]. These unique tokens can mark important locations or sample attributes in the input sequence, thus making the model better adapted to the target task.
- PSA: By introducing a pseudo-self-attention mechanism into the generation process, the model can selectively focus on different parts of the input text during the text generation process. This approach can effectively control specific attributes or styles of the generated text, such as sentiment, theme, or tone, leading to a more accurate text generation task.
- Fusion: this method realizes precise control of the attributes, style, and content of the generated text by combining information from multiple input modalities (e.g., text, image, speech, etc.) and introducing external control signals. This method can fuse multimodal details to make the generated text richer and more diverse, which can meet the needs of different application scenarios, such as emotion generation and theme transformation.
- Dynamic Prompting (DP) [49]: Dynamic cueing techniques control text generation by dynamically generating cues based on context. This approach utilizes contextual information to adjust prompt words in real time to make the generated text more coherent and aligned with expectations. It has the advantage of being adaptive and automatically adjusting the strategy according to different generation tasks.

4.3 Evaluation Metrics

This paper evaluates the following automated metrics for the target text:

- Perplexity (PPL): complexity PPL is used to evaluate language models and is often considered a proxy for the generation quality. All GPT -2-based models use a byte-level tokenization scheme (PPL-BPE), based on which the models' word-level complexity (PPL-Word) is also computed in this paper to compare them with previous models.
- ROUGE: The ROUGE score calculates the n-gram overlap between the test-generated text and the given target text. For completeness, ROUGE scores are computed for n-gram overlaps (ROUGE-1, ROUGE-2, and ROUGE-L) as well as F1 scores for each precision (P) and recall (R).
- BLEU: The BLEU score considers the degree of N-gram overlap between the generated text and the reference text. BLEU-4 is a form of the BLEU metric that evaluates the overlap of 4-grams (four consecutive words) and normalizes these matches. The BLEU value ranges from 0 to 1, with values closer to 1 indicating that the generated text is closer to the reference text.
- Manual Evaluation: Generated Readability, Relevance, and Completeness were manually scored (out of 10) in the model robustness experiment.

4.4 Parameter Settings

This paper uses the smallest publicly available version of GPT-2 as the model's decoder for computational efficiency. Specifically, it has L = 12 layers, H = 12 attention heads per layer, and a model dimensionality of d = 768 units, totaling 117 million parameters. In addition, To address the posterior collapse problem, we implement a cyclic annealing schedule by modifying the coefficient β before KL divergence in (2). Specifically, we have kept β close to zero in the first half of the cyclic schedule, linearly annealed β to 1 in the next one-fourth of the cyclic schedule and kept $\beta = 1$ in the remaining one-fourth of the cyclic schedule. Finally, In evaluation, we generate stories using the top-k top-p random sampling scheme with k = 100 and p = 0.95. Temperature smoothing technique is also applied with T = 0.95.

4.5 Threshold Selection

When structuring a text graph, the text similarity within the window needs to be compared with a threshold to dynamically adjust the window size to better capture the relationships between the texts. Therefore, the selection of the size of the threshold is critical. By choosing different thresholds for several experiments, the results are shown in Fig. 5. As can be seen from the figure, when the threshold is too small, almost all pairs of nodes in the graph are connected, resulting in the graph becoming too dense with nodes connected by a large number of irrelevant edges. These edges represent texts unrelated to the context, which can interfere with subsequent feature propagation and learning. At the same time, when the threshold is too large, the connectivity of the adjacency matrix decreases, and too low a connectivity may lead to isolated nodes or small isolated subgraphs, which prevents the GCN from efficiently propagating the features and, thus, from capturing the relationships between texts.

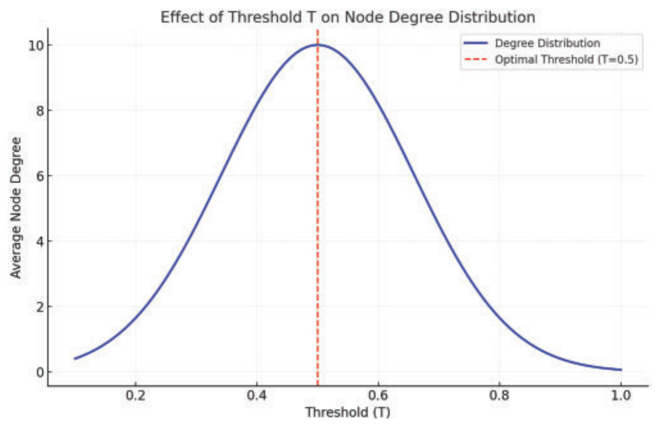

Figure 5: Between the threshold T and the node degree distribution of the adjacency matrix

4.6 Results

4.6.1 Pre-Experiment on VAE

To assess the effectiveness of transformer-based latent variable modeling, this study initially conducted pre-experiments using the VAE architecture on two small datasets. The prior followed a standard spherical Gaussian distribution, N(1, I). The VAE model performed pure unsupervised learning, where unlabeled linguistic text was encoded into a distributed representation and then decoded. The Arxiv and Yelp datasets were chosen for the pre-experiments. The Arxiv dataset was compiled by searching for topics in Arxiv, collecting article abstracts that matched the selected keywords: "Artificial Intelligence," "Computer Vision," and "Text Generation." The Yelp dataset, a public collection of restaurant reviews from the "Yelp" website, was also used. In this study, reviews with ratings above three were classified as positive, while those with ratings of three or below were considered negative.

4.6.2 Data Visualizations

Fig. 6 illustrates the posterior z of the text in the test dataset in a two-dimensional space using t-SNE. As can be seen from the figure, the model learns meaningful latent spaces and clusters the high-dimensional data based on the proximity between latent codes. In the Arxiv dataset, the "Artificial Intelligence" cluster sits between the "Computer Vision" and "Language Generation" clusters, which coincides with our understanding of these topics. This visualization shows the model's excellent representation learning capabilities.

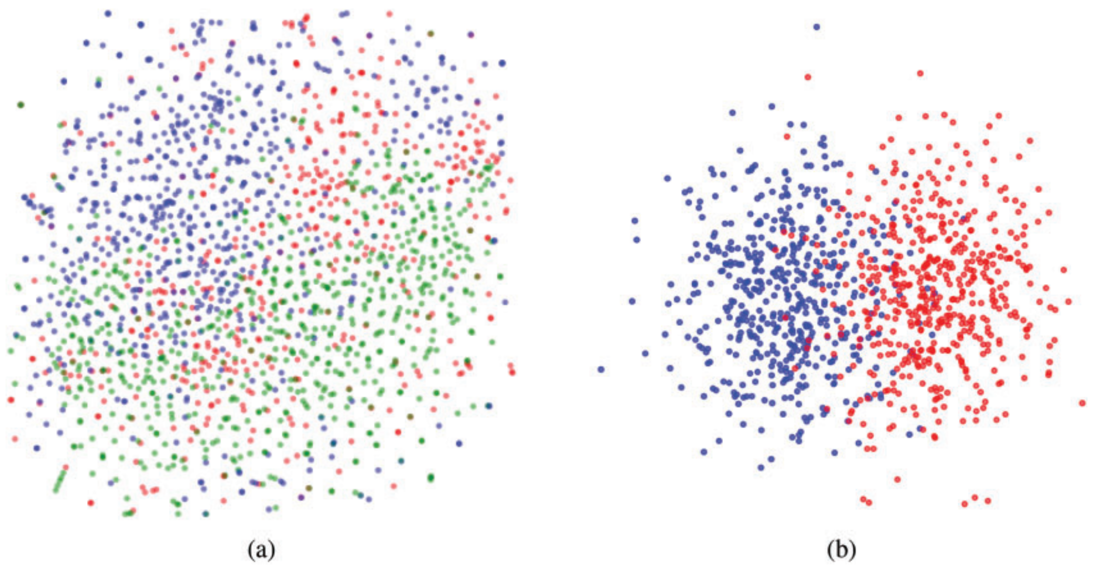

Figure 6: Pre-experiment results. (a) Arxiv: Topic are draw in different colors: red for artificial intelligence; blue for computer vision; green for language generation; (b) Yelp: Sentiment are draw in two colors: red for negative; blue for positive

Secondly, to illustrate the enhancement of text structure and coherence by graph structuring, this paper conducts comparative experiments on the Yelp dataset to compare the text structure and coherence between the original text representation and the textual relational representations after the text graph structuring process. The results are visualized by dimensionality reduction in t-SNE. As shown in Fig. 7, graph-structured texts (red dots) are more centrally clustered in 2D space and have firmer semantic consistency and coherence than original texts (blue dots). This suggests that the relationship between different text parts can be better captured through text graph structuring, thus improving the overall quality and controllability of the generated text.

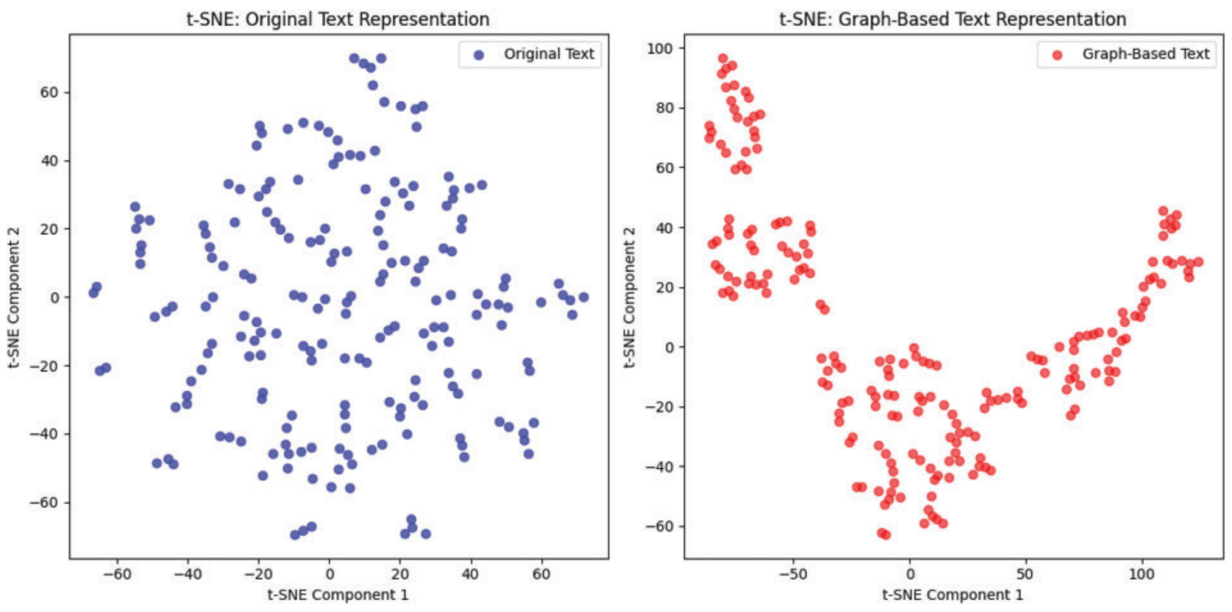

Figure 7: t-SNE visualization: comparison of textual relational representations of raw text and graph-structured text. The blue points represent the downscaled representation of the original text and the red points represent the downscaled representation of the graph-structured text

4.6.3 Comparative Experiments

Figs. 8 and 9 show the experimental results, which show that the word-level PPL and byte-level PPL of the GSPT-CVAE model are lower than those of other models, both from the beginning iteration to the 40,000th iteration and after the model has converged. In the ROUGE scoring metrics, the F1 scores of ROUGE-1, ROUGE-2, and ROUGE-L are higher than those of the traditional model, proving that the model in this paper has better performance, generation effectiveness, and controllability.

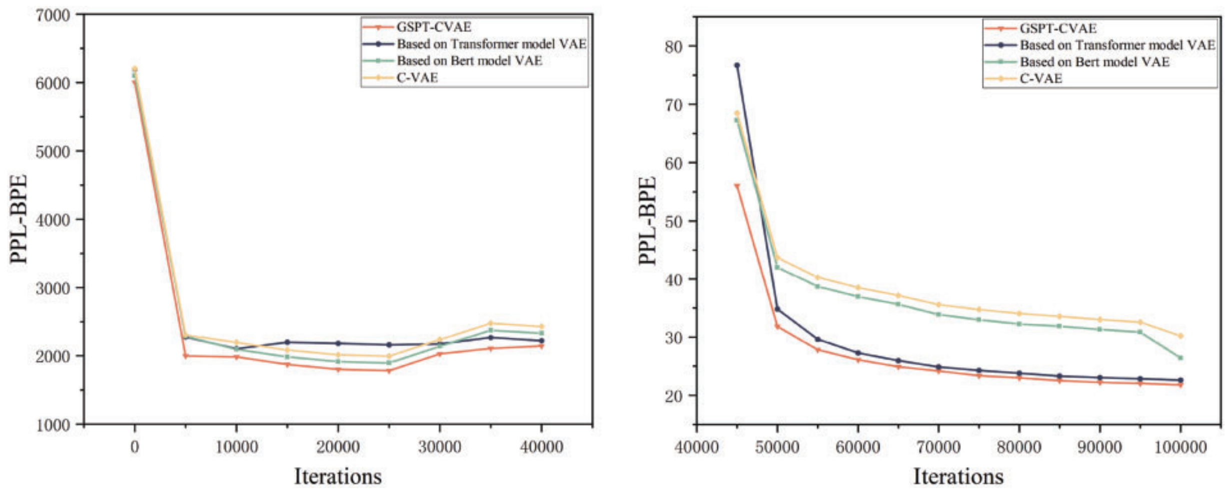

Figure 8: (Continued)

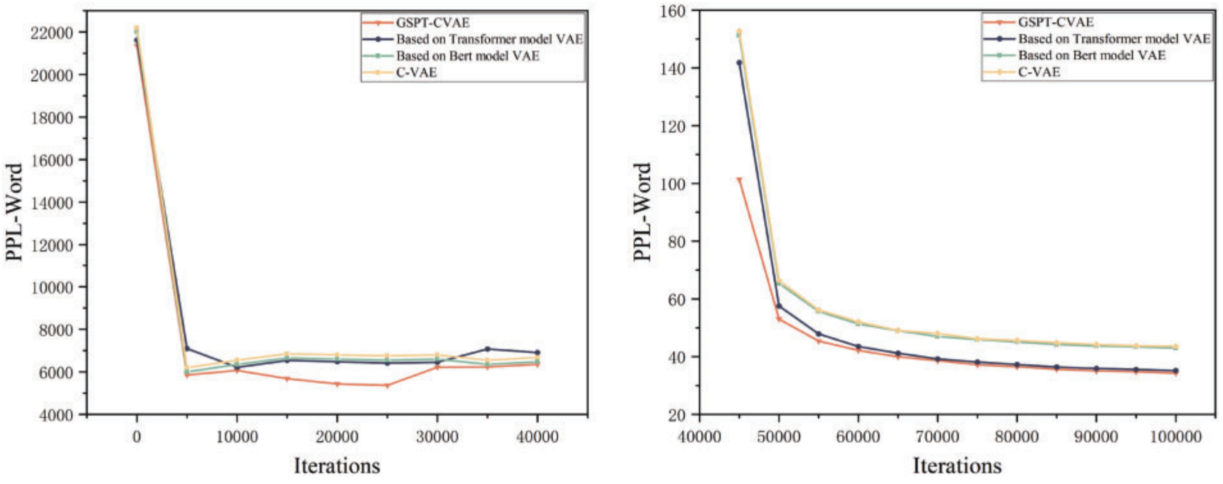

Figure 8: Comparison of experimental results between the GSPT-CVAE model and other models on the PPL-Word and PPL-BPE metrics

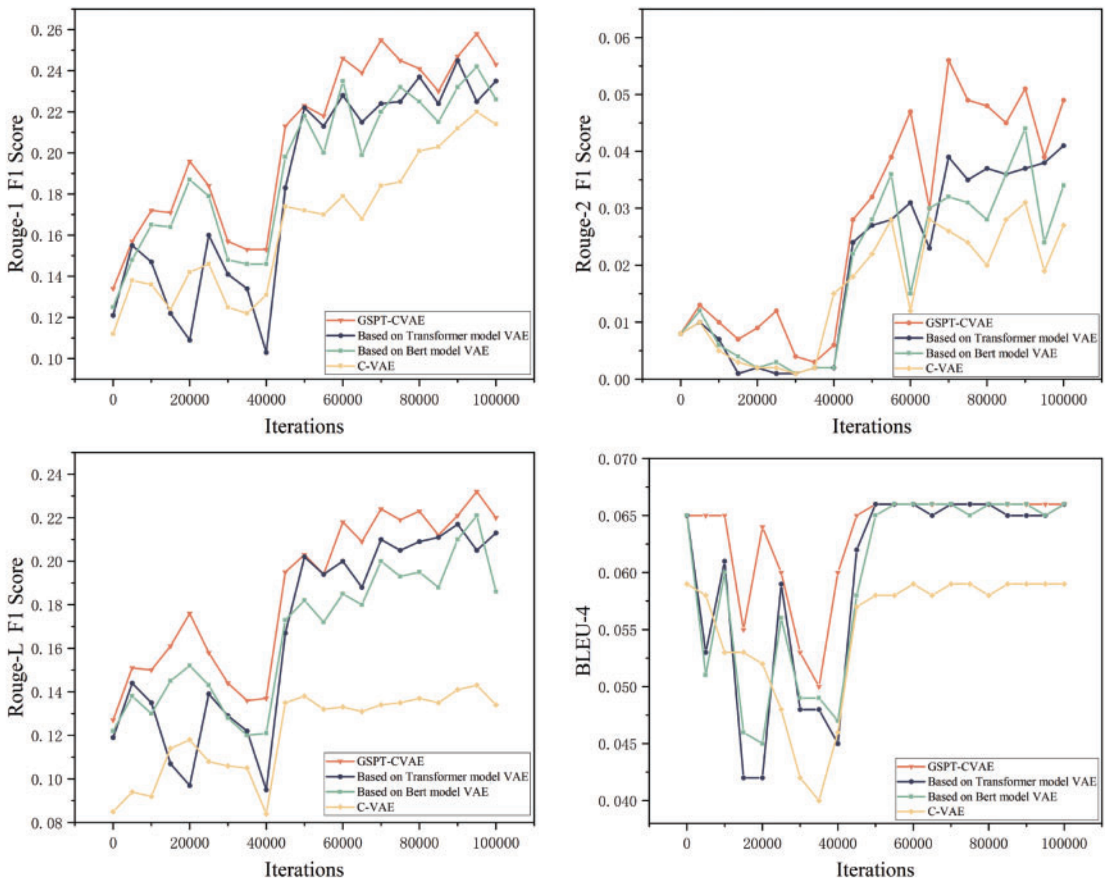

Figure 9: Comparison of experimental results between the GSPT-CVAE model and other models on the rouge score and BLEU-4 metrics

Meanwhile, the method used in the GSPT-CVAE model was compared with the comparative method chosen for this paper for various metrics, and the method used in the GSPT-CVAE model has better performance than other methods used for controlling text generation. Table 2 shows the experimental results of each method on the evaluation metrics, and Table 3 analyzes the advantages and disadvantages of the other methods and explains the superiority of GSPT-CVAE.

Table 2: Comparison of different methods for assessing metrics for controlled text generation

Methods	PPL-BPE	PPL-word	ROUGE-1 F1	ROUGE-2 F1	ROUGE-L F1
FIST	26.5	38.9	0.181	0.023	0.17
PSA	47.8	79.5	0.188	0.026	0.172
Fusion	–	36.0	0.223	0.038	0.206
DP	34.7	34.6	0.24	0.04	0.215
GSPT-CVAE	21.786	34.303	0.243	0.041	0.22

Table 3: Analysis of the strengths and weaknesses of the methods used in other models and the superiority of the methods used in GSPT-CVAE

Method	Features and advantages	Limitations	Improvements of GSPT-CVAE
FIST	- Guides the model to focus on specific tasks or samples through special tokens - Marks important positions or sample attributes	- Relies solely on tokens, making it difficult to capture complex semantic relationships - Limited to specific tasks, weak cross-task generalization	- Introduces structured text graphs to capture global semantic relationships - Stronger cross-task generalization capability generalization
PSA	- Uses pseudo self-attention to selectively focus on different parts of the input - Marks important positions or sample attributes	- Relies solely on tokens, making it difficult to capture complex semantic relationships - Limited to specific tasks, weak cross-task generalization	- Models global semantics using latent variables - Utilizes CVAE to achieve multi-attribute joint control
Fusion	- Integrates multi-modal inputs (e.g., text, images, speech), and generates richer and more diverse text in multi-modal scenarios - Provides precise control over text attributes, style, and content	- Performance drops in single-modal scenarios (e.g., text only) - Requires significant computational resources for multi-modal integration, reducing efficiency	- Uses latent variables and structured text graphs, eliminating the need for multi-modal input - Fully captures semantic relationships in single-modal scenarios
Dynamic Prompting (DP)	- Dynamically generates prompts based on context with adaptive capabilities - Ensures coherent text generation and adapts to various tasks	- Limited adaptability in complex control scenarios - Dynamic adjustments to context prompts may lead to instability in text control	- Stabilizes text control using latent variables and attention mechanisms - Provides efficient context modeling

Furthermore, we compare our model with T5, GPT-4, and Llama 2, representing state-of-the-art large language models. As seen in Figs. 10 and 11, GSPT-CVAE achieves a lower word-level PPL (34.303) than T5 (35.245) and Llama 2 (35.526), indicating its superior efficiency in controllable text generation. In terms of BLEU-4 scores, our model maintains a stable advantage, surpassing T5 (0.065), GPT-4 (0.064), and Llama 2 (0.063) with a final score of 0.066, demonstrating better consistency in text fluency. Regarding the ROUGE evaluation, GSPT-CVAE outperforms T5, GPT-4, and Llama 2 across all F1-score metrics. Precisely, in ROUGE-1, our model attains 0.243, compared to T5 (0.24), GPT-4 (0.237), and Llama 2 (0.235), confirming its superiority in capturing relevant textual features. Similarly, ROUGE-2 F1 scores show that GSPT-CVAE achieves 0.049, while T5, GPT-4, and Llama 2 obtain 0.047, 0.045, and 0.044, respectively, further reinforcing our model's ability to generate semantically rich and structurally coherent content. Lastly, for ROUGE-L F1, our model attains 0.22, which is higher than T5 (0.215), GPT-4 (0.212), and Llama 2 (0.21), highlighting its effectiveness in producing well-structured and meaningful long text.

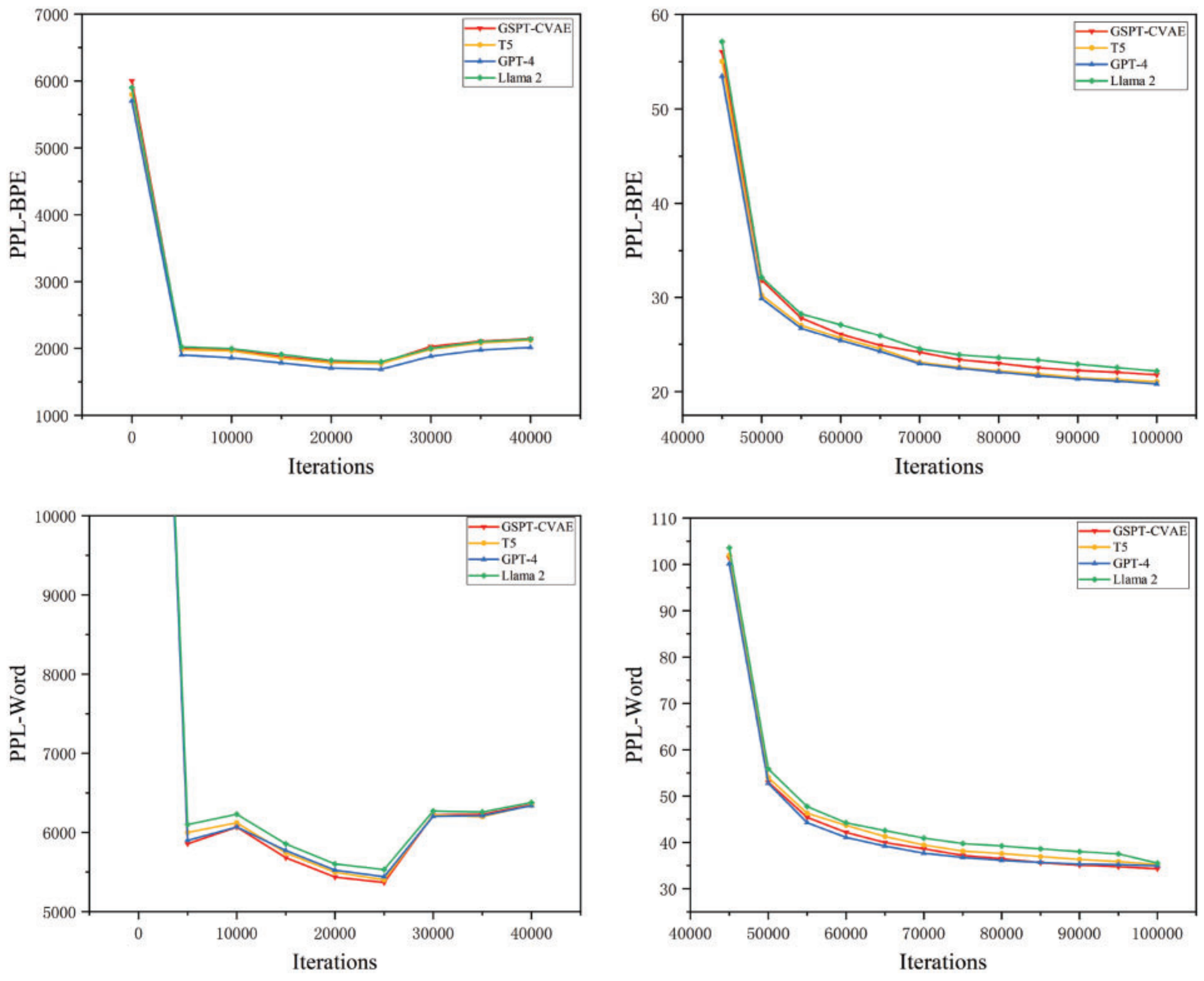

Figure 10: Comparison of experimental results between the GSPT-CVAE model and other models on the PPL-Word and PPL-BPE metrics

Finally, we conducted two more comparison experiments of the model on WritingPrompts and WikiPlots datasets in addition to the Arxiv dataset, and the results are shown in Table 4.

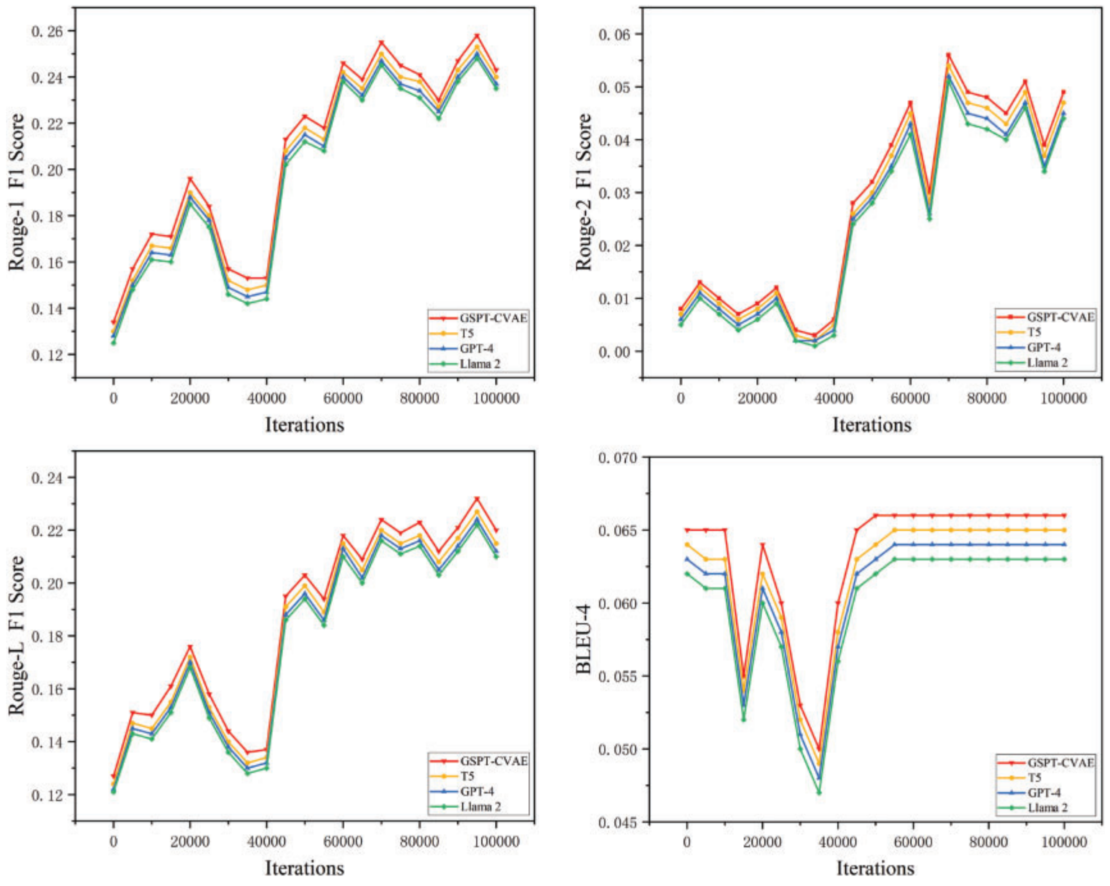

Figure 11: Comparison of experimental results between the GSPT-CVAE model and other models on the PPL-Word and PPL-BPE metrics

Table 4: Comparison of different methods for assessing metrics for controlled text generation

Models	PPL-BPE	PPL-word	ROUGE-1 F1	ROUGE-2 F1	ROUGE-L F1
WritingPrompts dataset					
T5	23.8	28.6	0.258	0.045	0.343
GPT-4	22.9	27.9	0.262	0.048	0.348
Llama 2	23.5	28.3	0.260	0.047	0.345
GSPT-CVAE	23.1	27.8	0.265	0.049	0.350
WikiPlots dataset					
T5	25.1	30.2	0.238	0.041	0.315
GPT-4	24.5	29.7	0.241	0.044	0.318
Llama 2	24.8	30.0	0.240	0.043	0.317
GSPT-CVAE	24.2	29.4	0.243	0.045	0.320

These results collectively demonstrate that GSPT-CVAE surpasses traditional controllable text generation models and maintains competitive performance against advanced large-scale models such as T5, GPT-4, and Llama 2, particularly in controllability, generation fluency, and coherence.

4.6.4 Robustness Experiment

To analyze the performance of GSPT-CVAE under adversarial conditions, this paper conducts robustness experiments on the model in terms of noisy inputs, very short cues, and domain shifts, respectively. Among them, noisy input is set to randomly insert noise (e.g., spelling mistakes, random symbols, missing words, etc.) into the input text to simulate dirty data in the real world; very short cue is set to input cues of extremely short length (less than five words); and domain transfer is set to go from the Arxiv scientific paper dataset to the CNN/DailyMail News dataset. The experimental results are shown in Tables 5–7.

Table 5: Noisy input test

Models	BLEU	ROUGE-L	PPL
GSPT-CVAE	18.4	42.5	26.1
T5	16.8	40.2	28.7
GPT-4	20.1	45.3	24.5

Table 6: Short prompt test

Models	Readability	Relevance	Completeness
GSPT-CVAE	8.3	7.9	7.5
T5	7.5	7.2	6.8
GPT-4	9.1	8.5	8.7

Table 7: Cross-domain test

Models	ROUGE-1	ROUGE-2 F1	ROUGE-L
GSPT-CVAE	39.2	19.5	35.1
T5	36.8	17.9	33.2
GPT-4	42.1	21.3	37.4

From the experimental results, when performing the input noise test, GSPT-CVAE outperforms T5 but is slightly lower than GPT-4. Its PPL value is relatively low, indicating it maintains good text fluency on noisy data. Due to the use of GCN structure to model global textual relations, even if part of the input information is corrupted, the model can still fill in the missing information by using the graph structure to reduce semantic distortion. In the future, data enhancement methods, such as spelling correction and noise simulation training, can be used to improve the model's robustness further. In the very short cue test, GSPT-CVAE can extend the short cue better and generate long text with certain contextual logic, which performs better than T5. At the same time, the information completeness score is lower than that of GPT-4, which may be because the latent variable relies on more contextual information. There is not enough information under the short cue. In the future, Few-shot Learning training can be added to adapt to very short cue scenarios or knowledge distillation can be introduced to enhance the short text generation capability of GSPT-CVAE by borrowing the high-quality text generated by GPT-4. Finally, in the cross-domain test, the generalization ability of GSPT-CVAE is more potent, which may be because the GCN structure can extract the global relationship, and even if the style of the text varies a lot, it can still maintain a certain degree of contextual consistency. The structure of the CVAE provides a more flexible control of latent variables, which is adaptable to the contexts of different domains.

4.6.5 Ablation Experiment

To verify that the methods used in the GSPT-CVAE model are indeed effective, this paper conducts ablation experiments on the three methods used in the model, i.e., the structured processing of the text map, the merging of the vector sequences, and the combination of the embedding and the potential attention computation. It visualizes the experimental results of the assessment metrics. As shown in Fig. 12, after eliminating the structured processing of the text map, both the F1 and BLEU scores in the ROUGE decreased significantly, indicating the effectiveness of the structured text map. In contrast, after eliminating the potential attention computation, the F1 score was lower than that of the GSPT-CVAE model except in some moments. In contrast, the BLEU score was ultimately lower than the GSPT-CVAE model's. Finally, after eliminating the merging layer of vector sequences Finally, after eliminating the vector sequence merging layer, the F1 scores are slightly higher than the original model in some moments. However, the overall scores are still significantly lower than the GSPT-CVAE model. The BLEU scores are lower than the GSPT-CVAE model in iterations and only gradually equal to the GSPT-CVAE model after the model converges.

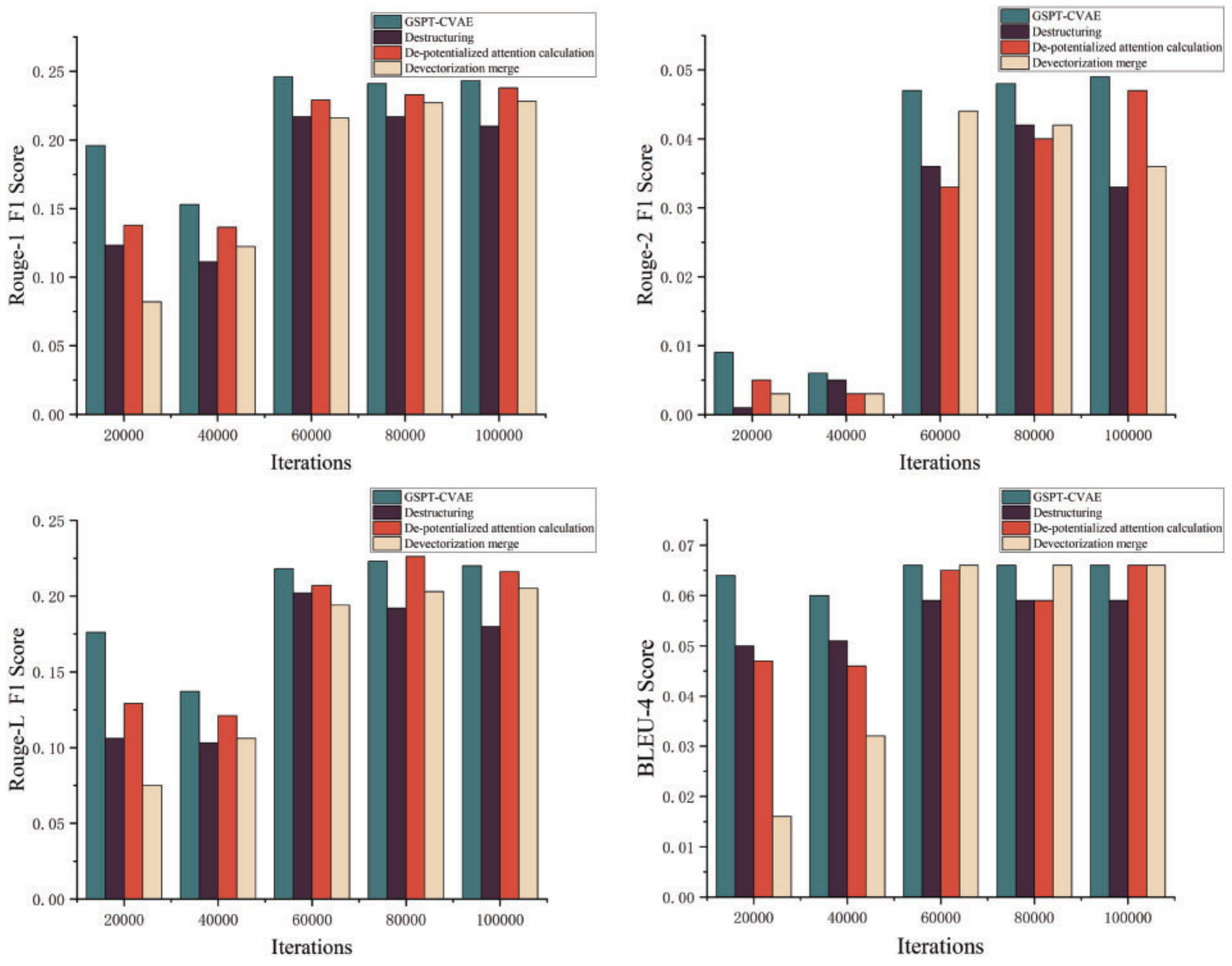

Figure 12: Results of ablation experiments of the three methods on each evaluation metrics

The results of the above ablation experiments show that all three methods used in the GSPT-CVAE model have enhanced the effectiveness and controllability of the model's text generation, which also verifies the superiority of the GSPT-CVAE model proposed in this paper in terms of controllable text generation.

4.6.6 Generated Cases

Table 8 shows the performance of the traditional CVAE model, T-CVAE model, Bert-based CVAE model, and the GSPT-CVAE model proposed in this paper in the actual text generation cases. From the table, it can be seen that all models can generate the corresponding text based on the conditions when the conditions are given. However, the text generated by the traditional CVAE model differs significantly from the target text. The length of the generated text is relatively short compared with other models. However, the text generated by the T-CVAE model is close to the target text in terms of sentence meaning and length, there are many sick sentences and logical errors in the text, and even grammatical errors; Bert-based CVAE model has a good overall performance in generating performance, but there are a few semantic ambiguities and other problems; while the GSPT-CVAE model proposed in this paper is the closest to the target text in terms of both text length and text quality, and there are no grammatical, logical, or semantic errors in the generated text. It is proved that the model has excellent long-text generation capability.

Table 8: Cases generated by different models on the Arxiv test dataset

Prompt text	Reactive programming in standard ML
Target text	A reactive system is a system that maintains a continuous interaction with its environment and is activated by receiving input events from the environment and generating output events in response. Modern programming languages specialized for programming such systems use a paradigm based on the concepts of transience and activation. We present a standard ML library that provides a basic prototype for programming reactive systems. The library is an underlying system upon which more complex reactive behaviors can be built, and it provides a convenient framework for prototypical extensions to existing reactive languages.
Generted text (CVAE)	We created a Python program similar to JavaScript but faster. It supports web and mobile apps, using HTML-like elements and functions for interactions. It can sort numbers and is memory-efficient, though it needs more testing for long tasks.
Generted text (T-CVAE)	We introduce a reactive ML language focused on dynamic values, adding more values and operations. Our aim is a reactive programming framework, demonstrated through an MFG tool case. We use a simplified predicate calculus model and briefly touch on type theory. Simplified reactive methods make this language practical for developers.
Generted text (Bert-CVAE)	We present a language based on ML's reactive paradigm, enabling explicit value assignments and dynamic updates. This reactive programming form in ML allows multi-value operations and uses specialized compilers. A case study illustrates its dynamic features, utilizing ML within simple predicate calculus to demonstrate functionality and provide new insights into programming languages.

(Continued)

Table 8 (continued)

Prompt text	Reactive programming in standard ML
Generted text (GSPT-CVAE)	We present a dynamic programming language based on ML's reactive programming paradigm, which extends ML by enabling explicit value assignments. This language supports multiple values, operations on values, and compiler usage. Using the MFG tool, we implement it in ML with a first-class predicate calculus and reformulate its type theory similarly to reactive programming. A case study demonstrates its implementation, yielding a type-theoretic description.

5 Conclusions

This paper presents a GSPT-CVAE model for controlled text generation. The model improves the effectiveness and controllability of text generation through three innovative approaches: text graph structuring, vector sequence merging, and combining embedding with latent attention computation. Unlike traditional controlled text generation, the model focuses on long text generation efforts through cueing methods, i.e., given an input cue text, the model generates long text within the domain of that cue text. The superiority of the GSPT-CVAE model and the veracity of the methodology used in the model are verified through experiments. The GSPT-CVAE model is of great significance in practical applications. It can significantly improve the efficiency and quality of content creation, especially for scenarios that require rich expression and diversified generation, such as literary creation and news writing. The model can support personalized teaching by generating personalized learning materials according to different topics and objectives in education. In addition, the model is widely used for automatic report generation, for example, in business, healthcare, and scientific research, where it can generate structured, logical, and long text reports based on input prompts, thus significantly reducing manual editing time. These application scenarios demonstrate the potential of the GSPT-CVAE model as an indispensable tool for text-generation tasks. The model can be deployed as a microservices-based API, allowing developers to integrate it into existing content management platforms, chatbots, and automated writing assistants. By leveraging a distributed inference architecture (e.g., deployed on a cloud-based GPU cluster or deployed at the edge using model quantization), GSPT-CVAE can efficiently process real-time text generation requests. Additionally, a Retrieval Augmented Generation (RAG) pipeline can be added to enable the model to access external knowledge bases, thus improving factual accuracy and reducing illusions. Future work will focus on optimizing the inference speed of the model, improving its robustness to brief prompts, and developing interactive fine-tuning mechanisms that dynamically incorporate user feedback to improve the quality of the output. These advances will further solidify the reliability and scalability of GSPT-CVAE for real-world text generation tasks. However, the model still has some limitations, such as the sparse representation of graphs obtained by structuring text graphs when the input prompts are too short. This leads to problems such as the model results may not be optimal. Therefore, there is still a long way to go in the research of controlled text generation, and there are many methods or new models that can be improved or optimized in the future, and more efforts need to be invested in the research.

Acknowledgement: None.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: Jun Tu; data collection, analysis, interpretation of results and draft manuscript preparation: Tian Zhao; manuscript guidance: Puzheng Quan, Ruisheng Xiong. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: Data openly available in a public repository. The data that support the findings of this study are openly available in github at https://github.com/gcunhase/ArXivAbsTitleDataset; https://github.com/seanreed1111/yelp-reviews (accessed on 10 April 2025).

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

1. Zhang H, Song H, Li S, Zhou M, Song D. A survey of controllable text generation using transformer-based pre-trained language models. ACM Comput Surv. 2023;56(3):1–37. doi:10.1145/3676955.
2. Prabhumoye S, Black AW, Salakhutdinov R. Exploring controllable text generation techniques. In: Proceedings of the 28th International Conference on Computational Linguistics. Barcelona, Spain (Online); 2020. p. 1–14.
3. Liu P, Yuan W, Fu J, Jiang Z, Hayashi H, Neubig G. Pre-train, prompt, and predict: a systematic survey of prompting methods in natural language processing. ACM Comput Surv. 2023;55(9):1–35. doi:10.1145/3560815.
4. Zheng X, Lin H, Han X, Sun L. Toward unified controllable text generation via regular expression instruction. In: Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers). Nusa Dua, Bali: Association for Computational Linguistics; 2023. p. 1–14.
5. Han J, Wang Q, Zhang L, Chen W, Song Y, Mao Z. Text style transfer with contrastive transfer pattern mining. In: Rogers A, Boyd-Graber J, Okazaki N, editors. Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Toronto, ON, Canada: Association for Computational Linguistics; 2023. p. 7914–27.
6. He X. Parallel refinements for lexically constrained text generation with BART. In: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing. Online and Punta Cana, Dominican Republic; 2021. p. 8653–66.
7. Cui Y, Che W, Liu T, Qin B, Wang S, Hu G. Revisiting pre-trained models for chinese natural language processing. In: Cohn T, He Y, Liu Y, editors. Findings of the Association for Computational Linguistics: EMNLP 2020. Online; 2020. p. 657–68.
8. Ouyang L, Wu J, Jiang X, Almeida D, Wainwright C, Mishkin P, et al. Training language models to follow instructions with human feedback. In: Koyejo S, Mohamed S, Agarwal A, Belgrave D, Cho K, Oh A, editors. Advances in neural information processing systems. Vol. 35. Newry, UK: Curran Associates, Inc.; 2022. p. 27730–44.
9. Zhou W, Jiang YE, Wilcox E, Cotterell R, Sachan M. Controlled text generation with natural language instructions. In: Proceedings of the 40th International Conference on Machine Learning; 2023. Vol. 202, p. 42602–13
10. Qian J, Dong L, Shen Y, Wei F, Chen W. Controllable natural language generation with contrastive prefixes. In: Findings of the Association for Computational Linguistics: ACL 2022; 2022; Dublin, Ireland. p. 2912–24.
11. Gu Y, Feng X, Ma S, Zhang L, Gong H, Qin B. A distributional lens for multi-aspect controllable text generation. In: Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing; 2022; Abu Dhabi, United Arab Emirates. p. 1023–43.
12. Zou X, Yin D, Zhong Q, Yang H, Yang Z, Tang J. Controllable generation from pre-trained language models via inverse prompting. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining: KDD '21; 2021; New York, NY, USA. p. 2450–60. doi:10.1145/3447548.3467418.
13. Pascual D, Egressy B, Meister C, Cotterell R, Wattenhofer R. A plug-and-play method for controlled text generation. In: Moens MF, Huang X, Specia L, Wen-tau Yih S, editors. Findings of the Association for Computational Linguistics: EMNLP 2021. Punta Cana, Dominican Republic; 2021. p. 3973–97.

14. Yang K, Klein D. FUDGE: controlled text generation with future discriminators. In: Toutanova K, Rumshisky A, Zettlemoyer L, Hakkani-Tur D, Beltagy I, Bethard S, et al., editors. Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Online: Association for Computational Linguistics; 2021. p. 3511–35.
15. Gu Y, Feng X, Ma S, Wu J, Gong H, Qin B. Improving controllable text generation with position-aware weighted decoding. In: Muresan S, Nakov P, Villavicencio A, editors. Findings of the Association for Computational Linguistics: ACL 2022. Dublin, Ireland: Association for Computational Linguistics; 2022. p. 3449–67.
16. Chen J, Wu Y, Jia C, Zheng H, Huang G. Customizable text generation via conditional text generative adversarial network. Neurocomputing. 2020;416(1):125–35. doi:10.1016/j.neucom.2018.12.092.
17. Welleck S, Kulikov I, Roller S, Dinan E, Cho K, Weston J. Neural text generation with unlikelihood training. In: International Conference on Learning Representations; 2020. p. 1–18.
18. Su Y, Lan T, Wang Y, Yogatama D, Kong L, Collier N. A contrastive framework for neural text generation. In: Koyejo S, Mohamed S, Agarwal A, Belgrave D, Cho K, Oh A, editors. Advances in neural information processing systems. Vol. 35. Newry, UK: Curran Associates, Inc.; 2022. p. 21548–61.
19. Shu L, Papangelis A, Wang YC, Tur G, Xu H, Feizollahi Z, et al. Controllable text generation with focused variation. In: Cohn T, He Y, Liu Y, editors. Findings of the Association for Computational Linguistics: EMNLP 2020. Online: Association for Computational Linguistics; 2020. p. 3805–17.
20. Chen Y, Gröner H, Zarrieß S, Eger S. Evaluating diversity in automatic poetry generation. In: Al-Onaizan Y, Bansal M, Chen YN, editors. Proceedings of the 2024 conference on empirical methods in natural language processing. Miami, FL, USA: Association for Computational Linguistics; 2024. p. 19671–92.
21. Li X, Thickstun J, Gulrajani I, Liang PS, Hashimoto TB. Diffusion-LM improves controllable text generation. In: Koyejo S, Mohamed S, Agarwal A, Belgrave D, Cho K, Oh A, editors. Advances in neural information processing systems. Vol. 35. Newry, UK: Curran Associates, Inc.; 2022. p. 4328–43.
22. Lovelace J, Kishore V, Wan C, Shekhtman E, Weinberger KQ. Latent diffusion for language generation. In: Oh A, Naumann T, Globerson A, Saenko K, Hardt M, Levine S, editors. Advances in neural information processing systems. Vol. 36. Newry, UK: Curran Associates, Inc.; 2023. p. 56998–7025.
23. Liang X, Tang Z, Li J, Zhang M. Open-ended long text generation via masked language modeling. In: Rogers A, Boyd-Graber J, Okazaki N, editors. Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Toronto, ON, Canada: Association for Computational Linguistics; 2023. p. 223–41.
24. Li Q, Li P, Bi W, Ren Z, Lai Y, Kong L. Event transition planning for open-ended text generation. In: Muresan S, Nakov P, Villavicencio A, editors. Findings of the Association for Computational Linguistics: ACL 2022. Dublin, Ireland: Association for Computational Linguistics; 2022 May 22–27. p. 3412–26. doi:10.18653/v1/2022.findings-acl.269.
25. Mu F, Li W. Enhancing text generation via multi-level knowledge aware reasoning. In: International Joint Conference on Artificial Intelligence; 2022; Vienna, Austria. p. 4310–6.
26. Hu Z, Chan HP, Liu J, Xiao X, Wu H, Huang L. PLANET: dynamic content planning in autoregressive transformers for long-form text generation. In: Muresan S, Nakov P, Villavicencio A, editors. Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Dublin, Ireland: Association for Computational Linguistics; 2022. p. 2288–305.
27. Krause B, Gotmare AD, McCann B, Keskar NS, Joty S, Socher R, et al. GeDi: generative discriminator guided sequence generation. In: Moens MF, Huang X, Specia L, Wen-tau Yih S, editors. Findings of the Association for Computational Linguistics: EMNLP 2021. Punta Cana, Dominican Republic: Association for Computational Linguistics; 2021. p. 4929–52.
28. Guan J, Huang F, Zhao Z, Zhu X, Huang M. A knowledge-enhanced pretraining model for commonsense story generation. Trans Assoc Comput Linguist. 2020;8(5):93–108. doi:10.1162/tacl_a_00302.

29. Guan J, Mao X, Fan C, Liu Z, Ding W, Huang M. Long text generation by modeling sentence-level and discourse-level coherence. In: Zong C, Xia F, Li W, Navigli R, editors. Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers); 2021 Aug 1–6. Virtual Event: Association for Computational Linguistics; 2021. p. 6379–93.
30. Xu P, Patwary M, Shoeybi M, Puri R, Fung P, Anandkumar A, et al. Controllable story generation with external knowledge using large-scale language models. In: Conference on Empirical Methods in Natural Language Processing 2020. Online; 2020. p. 2831–45.
31. Lee J, Stevens N, Han SC, Song M. A survey of large language models in finance (FinLLMs). Neural Comput Appl. 2024;33(240):1877. doi:10.1007/s00521-024-10495-6.
32. Liang X, Song S, Niu S, Li Z, Xiong F, Tang B, et al. UHGEval: benchmarking the hallucination of chinese large language models via unconstrained generation. In: Ku LW, Martins A, Srikumar V, editors. Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Bangkok, Thailand: Association for Computational Linguistics; 2024. p. 5266–93.
33. Bartsch H, Jorgensen O, Rosati D, Hoelscher-Obermaier J, Pfau J. Self-consistency of large language models under ambiguity. In: Belinkov Y, Hao S, Jumelet J, Kim N, McCarthy A, Mohebbi H, editors. Proceedings of the 6th blackboxnlp workshop: analyzing and interpreting neural networks for NLP. Singapore: Association for Computational Linguistics; 2023. p. 89–105.
34. Havaldar S, Singhal B, Rai S, Liu L, Guntuku SC, Ungar L. Multilingual language models are not multicultural: a case study in emotion. In: Barnes J, De Clercq O, Klinger R, editors. Proceedings of the 13th Workshop on Computational Approaches to Subjectivity, Sentiment, & Social Media Analysis. Toronto, ON, Canada: Association for Computational Linguistics; 2023. p. 202–14.
35. Liu Z, Yang K, Zhang T, Xie Q, Yu Z, Ananiadou S. EmoLLMs: a series of emotional large language models and annotation tools for comprehensive affective analysis. In: KDD'24: Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining; 2024; Barcelona, Spain. p. 5487–96.
36. Hua X, Wang L. PAIR: planning and iterative refinement in pre-trained transformers for long text generation. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). Online; 2020. p. 781–93.
37. Kitaev N, Kaiser L, Levskaya A. Reformer: the efficient transformer. In: ICLR 2020: The Eighth International Conference on Learning Representations. Online; 2020. p. 1–12.
38. Qin G, Feng Y, Van Durme B. The NLP task effectiveness of long-range transformers. In: Vlachos A, Augenstein I, editors. Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics. Dubrovnik, Croatia: Association for Computational Linguistics; 2023. p. 3774–90.
39. Zhang Y, Carvalho D, Valentino M, Pratt-Hartmann I, Freitas A. Improving semantic control in discrete latent spaces with transformer quantized variational autoencoders. In: Graham Y, Purver M, editors. Findings of the Association for Computational Linguistics: EACL 2024. St. Julian's, Malta: Association for Computational Linguistics; 2024. p. 1434–50.
40. Yang X, Peynetti E, Meerman V, Tanner C. What GPT knows about who is who. In: Tafreshi S, Sedoc J, Rogers A, Drozd A, Rumshisky A, Akula A, editors. Proceedings of the Third Workshop on Insights from Negative Results in NLP. Dublin, Ireland: Association for Computational Linguistics; 2022. p. 75–81.
41. Lei Y, Huang R, Wang L, Beauchamp N. Sentence-level media bias analysis informed by discourse structures. In: Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing; 2022; Abu Dhabi, United Arab Emirates. p. 10040–50.
42. Cai J, Ahmed SR, Bonn J, Wright-Bettner K, Palmer M, Martin JH. CAMRA: copilot for AMR annotation. In: Feng Y, Lefever E, editors. Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. Singapore: Association for Computational Linguistics; 2023. p. 381–8.
43. Wang F, Li F, Fei H, Li J, Wu S, Su F, et al. Entity-centered cross-document relation extraction. In: Goldberg Y, Kozareva Z, Zhang Y, editors. Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics; 2022. p. 9871–81.

44. Sun X, Sun Z, Meng Y, Li J, Fan C. Summarize, outline, and elaborate: long-text generation via hierarchical supervision from extractive summaries. In: Calzolari N, Huang CR, Kim H, Pustejovsky J, Wanner L, Choi KS et al., editors. Proceedings of the 29th International Conference on Computational Linguistics. Gyeongju, Republic of Korea: International Committee on Computational Linguistics; 2022. p. 6392–402.
45. Cavalin PR, Vasconcelos M, Grave M, Pinhanez CS, Ribeiro VHA. From disjoint sets to parallel data to train seq2seq models for sentiment transfer. In: Findings of the Association for Computational Linguistics: EMNLP 2020. Online; 2020. p. 689–98.
46. Wilmot D, Keller F. Modelling suspense in short stories as uncertainty reduction over neural representation. In: Jurafsky D, Chai J, Schluter N, Tetreault J, editors. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. Online: Association for Computational Linguistics; 2020. p. 1763–88.
47. Yang E, Liu M, Xiong D, Zhang Y, Chen Y, Xu J. Long text generation with topic-aware discrete latent variable model. In: Goldberg Y, Kozareva Z, Zhang Y, editors. Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing. United Arab Emirates: Association for Computational Linguistics; 2022. p. 8100–7.
48. Song Y, Liu Z, Bi W, Yan R, Zhang M. Learning to customize model structures for few-shot dialogue generation tasks. In: Jurafsky D, Chai J, Schluter N, Tetreault J, editors. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. Online: Association for Computational Linguistics; 2020. p. 5832–41.
49. Swamy S, Tabari N, Chen C, Gangadharaiah R. Contextual dynamic prompting for response generation in task-oriented dialog Systems. In: Vlachos A, Augenstein I, editors. Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics. Dubrovnik, Croatia: Association for Computational Linguistics; 2023. p. 3102–11.