entropy

MDPI

Article

A Transformer-Based Hierarchical Variational AutoEncoder Combined Hidden Markov Model for Long Text Generation

Kun Zhao, Hongwei Ding, Kai Ye and Xiaohui Cui *

Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China; zhaokun@whu.edu.cn (K.Z.); hwding@whu.edu.cn (H.D.); kai_ye@whu.edu.cn (K.Y.)
* Correspondence: xcui@whu.edu.cn

Abstract: The Variational AutoEncoder (VAE) has made significant progress in text generation, but it focused on short text (always a sentence). Long texts consist of multiple sentences. There is a particular relationship between each sentence, especially between the latent variables that control the generation of the sentences. The relationships between these latent variables help in generating continuous and logically connected long texts. There exist very few studies on the relationships between these latent variables. We proposed a method for combining the Transformer-Based Hierarchical Variational AutoEncoder and Hidden Markov Model (HT-HVAE) to learn multiple hierarchical latent variables and their relationships. This application improves long text generation. We use a hierarchical Transformer encoder to encode the long texts in order to obtain better hierarchical information of the long text. HT-HVAE's generation network uses HMM to learn the relationship between latent variables. We also proposed a method for calculating the perplexity for the multiple hierarchical latent variable structure. The experimental results show that our model is more effective in the dataset with strong logic, alleviates the notorious posterior collapse problem, and generates more continuous and logically connected long text.

Keywords: Variational AutoEncoder; text generation; Hidden Markov Model; Transformer; latent variables

Citation: Zhao, K.; Ding, H.; Ye, K.; Cui, X. A Transformer-Based Hierarchical Variational AutoEncoder Combined Hidden Markov Model for Long Text Generation. *Entropy* **2021**, *23*, 1277. https://doi.org/10.3390/e23101277

Academic Editors: Salim Lahmiri and Sotiris Kotsiantis

Received: 2 September 2021
Accepted: 27 September 2021
Published: 29 September 2021

1. Introduction

Natural Language Processing (NLP) has achieved great progress in these years, including text classification [1–4], text generation [5–10], and sentiment analysis [11–13]. At the same time, deep recurrent belief [14] and capsule networks [15–17] also have new challenges in the field of NLP.

Deep neural networks [18] have achieved great success in the so-called neural generation of text [5–10]. In the field of text generation, the most common is the sequence-to-sequence(seq2seq) model [19], which consists of an encoder and a decoder. The encoder encodes the text into a fixed vector, and the decoder decodes the corresponding text according to this vector. However, since the encoder in seq2seq encodes the text into a fixed vector, the text generated by the decoder is too monotonous and cannot generate diverse text. The Variational AutoEncoder (VAE) [20,21] randomly samples the encoded representation vector from the hidden space, and the decoder can generate real and novel text based on the latent variables. VAE provides a tractable method to train generative models of latent variables. In NLP, latent variables may represent higher-level meanings of sentences, such as semantics or topics. These latent variables control the generation of text [22]. Compared with the traditional seq2seq model, the latent variables in the VAE are considered to make the model more powerful.

In Natural Language Generation, Long Short Term Memory networks (LSTM) [23] and Transformer structures [24] are mostly used in the text generation field. The Transformer completely uses the attention mechanism. There are many Transformer-based models,

including Bert [25], UniLM [26], BART [27], T5 [28], and GPT-2 [8]. Most VAEs use the LSTM structure for text generation [22,29–31], and very few VAEs use the Transformer structure [32–34].

At present, VAE is mainly used to generate short text (less than 20 words or one sentence) in text generation, and few people pay attention to the generation of long text. For long texts, words are often composed of sentences, and sentences are composed of text. Each sentence may contain different semantics and other latent variables information, and the latent variables of each sentence may also have a progressive relationship. However, few people focused on it. Therefore, in order to improve long text generation, we proposed a new method for combining the Transformer-Based Hierarchical Variational AutoEncoder and Hidden Markov Model (HT-HVAE) to learn multiple hierarchical latent variables and their relationships.

Humans think about their purpose before writing, which is the meaning of this article. After that, an individual will write by progressing from sentence to sentence. The act of writing these sentences needs to meet the writing purpose, and the previous sentence simultaneously determines the generation of the following sentence. However, the purpose of writing is not observable. For example, when reading a long text, we can only know the author's writing purpose if we read the entire text. In addition, each sentence in the text has its own semantics or topics. These are also latent variables, but these latent variables, such as semantics or topics, cannot exist without the purpose of writing. It is these latent variables that control the generation of the entire long text. We call the latent variables that control the generation of all sentences similar to the purpose of writing as global latent variables, and the latent variables that specifically control the generation of each sentence (such as semantics or topics) are called local latent variables. In addition, the current local latent variables are also related to the previous local latent variables (for example, semantic conversion, etc.).

We used a Hierarchical Transformer to encode long text for the encoder of HT-HVAE. We used a word-level Transformer to encode every sentence (we denoted it as sentence-code). All sentence-code passes sentence-level transformers to acquire text representations (we denote the represent as text-code). The sentence-code will be used to learn local latent variables and text-code to learn global latent variables. The decoder also applies hierarchical structure, which combines Gated Recurrent Unit (GRU) [35] with GPT-2, to reconstruct texts. The input of GRU includes local latent variables, and we denote GRU's output as plan-vectors. The plan-vector will be copied in n copies, where n is the length of the sentence as the input of GPT-2 to generate the sentence.

We always maximize Evidence Lower Bound (ELBO) in the training process of VAE. Thus, we can use ELBO to approximately calculate the perplexity (PPL) of the language model. However, ELBO always makes the calculated PPL inaccurate. The Importance Weighted AutoEncoder (IWAE) [36] proposed a method that PPL can calculate a single latent variable as accurately as possible. Many researchers used this method to calculate PPL as accurately as possible [33]. However, this method is mainly for single latent variables, and HT-HVAE has multiple latent variables. We adopted the idea of IWAE and expanded the method of accurately calculating PPL in the case of multi-level latent variables.

In summary, our work mainly explores the following:

1. In order to further generate real long texts, we propose a new model that combines VAE, HMM, and hierarchical structure to learn one global latent variable and multiple local latent variables.
2. The global latent variables (from the entire text) learned by our method control the local latent variables (from each sentence), and the local latent variables at the sentence level also have dependencies. Moreover, this is similar to the way humans write.
3. According to IWAE, we extend this method for the hierarchical latent variable in order to obtain a more accurate PPL.
4. Experiments prove that our model alleviates the notorious posterior collapse problem in VAE.

The topic in the first section is the Introduction. The following sections are structured as follows: Section 2 introduces related work, Section 3 introduces background knowledge, Section 4 introduces the methods we use, Section 5 introduces experimental results, and Section 6 is the conclusion.

2. Related Work

VAE has made significant progress in text generation [22,30,32–34,37–43]. In this reference [22], researchers used VAE for text generation for the first time. Researchers proved that learning higher-level text representations (such as semantics or topics) is possible through LSTM-based latent variables generative models, and these higher-level representations control text generation. Due to the sampling process of latent variables, the diversity of the generated text is better. [44].

However, most of VAE's work focuses on the generation of short texts (mostly one sentence), and few people use VAE to generate long texts. Shen et al. [29] proposes a new hierarchical structure named hVAE to learn a generative model of two-layer latent variables to generate long text: the global latent variables of the entire text and the local latent variables at the sentence level. However, hVAE still has some problems: (1) There is only one sentence-level local latent variable, and it does not learn the local latent variables of each sentence. (2) It does not explore the dependency relationship between sentence-level local latent variables, for example, semantic transformation. (3) Its framework does not deviate from the LSTM structure.

Recently, the Transformer has achieved great success in the field of NLP and has surpassed LSTM, and the most famous one in text generation is GPT-2. However, the attention mechanism of GPT-2 only pays attention to the relationship between words and does not pay attention to the relationship between sentences. In addition, few people combine GPT-2 or Transformer and VAE. Liu et al. [32] combined Transformer and VAE for text generation for the first time; Wang et al. [34] used Transformer and CVAE for story complement tasks; Li et al. [33] used Bert as the encoder for VAE, and GPT-2 is used as a VAE decoder to train a large pretraining model to solve various tasks of natural language processing. However, Optimus [33] only uses the standard VAE method; that is, it only learns one hidden variable. Therefore, its application scenario is still short text, which is not suitable for long text generation. In addition, even though Optimus provides latent variable information in the decoder part, it still fails to make GPT-2 pay attention to the connection between the latent variables at the sentence level.

By considering the shortcomings of hVAE, GPT-2, and Optimus, we proposed HT-HVAE to learn a global latent variable (from the entire text) and multiple local latent variables (from each sentence). Furthermore, we propose it to learn the dependence between global latent variables and local latent variables and the connection between local latent variables. In addition, in the decoder part, we input local latent variables into GPT-2 so that GPT-2 pays attention to the relationship between words and the relationship between sentence-level latent variables.

However, text VAE also has a notorious posterior collapse problem. In recent years, several methods have been working to alleviate this problem, including different Kullback–Leibler (KL) annealing/thresholding schemes [22,45–47], decoder architectures [38,48], auxiliary loss [44], semi amortized inference [41], aggressive encoder training schedule [49], and flexible posterior [50]. However, none of them help to improve the perplexity compared to a plain neural language model. In this article, because the local latent variable sent to the decoder not only contains the information of the global latent variable but also contains the information of the previous local latent variable, it makes the prior distribution function of the local latent variable more flexible. Therefore, the decoder contains more local latent variable information, which alleviates the problem of posterior collapse.

3. Background

3.1. VAE

As shown in Figure 1, VAE mainly includes inference networks and generative networks. In NLP, the inference network mainly represents sentences as low-dimensional latent variables, and the generative network uses this latent variable to generate sentences. In fact, VAE can be regarded as a regularized version of the AutoEncoder (AE) [18], but the low dimension of the sentence is not a fixed code, unlike AE, and is sampled from a probability distribution.

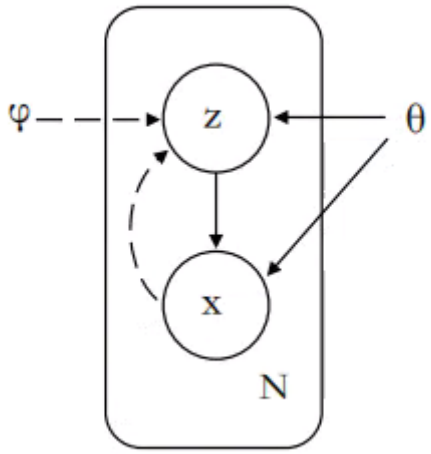

Figure 1. VAE. The solid line represents the generative network, and the dashed line represents the inference network [20].

Suppose x is the observable data obtained by sampling, such as a sentence, and z is a low-dimensional latent variable, such as semantics or topics. The joint probability density function of this generative model can be decomposed into the following.

$$p(x, z|\theta) = p(z|\theta)p(x|z;\theta) \tag{1}$$

$p(z|\theta)$ is the prior distribution of the latent variable z, $p(x|z;\theta)$ is the conditional probability distribution of x when z is known, and θ is the parameter of the two probability distributions. We assume that $p(z|\theta)$ and $p(x|z;\theta)$ are a known distribution family, such as a normal distribution. In VAE, we used two neural networks to fit these two distribution families separately. In fact, we assume the standard normal distribution of $p(z)$ and use a network to learn the probability distribution $p(x|z;\theta)$. This network is called the generative network, also called decoder, and its input is a latent variable z while the output is the observable variable x.

The purpose of the inference network is to learn the probability distribution $q(z|x;\phi)$, which is used to approximate the true posterior distribution $p(z|x;\theta)$. The input of the inference network is x, and the output is the probability distribution $q(z|x,\theta)$. Similarly, we assume that $q(z|x;\theta)$ is Gaussian distribution.

It has been proved in [20] that the objective function of VAE is $ELBO$, which is composed of the reconstruction term and the KL divergency.

$$ELBO = E_{q(z|x;\phi)} log p(x|z;\theta) - KL(q(z|x;\theta)||p(z;\theta)) \tag{2}$$

The first term in $ELBO$ is called the reconstruction loss, which is the same as the AE, and the second term is KL loss, which can be regarded as a regularized item. The KL loss forces the distribution of the hidden variable z to approximate $p(z;\theta)$ as much as possible and to sample from it in order to generate real text.

When VAE is used in the field of text generation, most of the inference network and the generative network utilize the LSTM structure. Due to the strong decoding ability of LSTM, x can be generated without z during the generation process, so the notorious "posterior collapse" problem will occur. In HT-HVAE, we use Transformer as the inference network and the generative network. For single-layer latent variables, the problem of posterior collapse will also occur. However, we have learned two layers of latent variables, namely the global latent variable z_t of the entire text and the local latent variable z_i of each sentence.

Moreover, for each sentence, the prior latent variables z_i are related to prior hidden variables z_{i-1} of the previous sentence. The prior distribution of z_i is no longer a simple standard diagonal Gaussian distribution, rendering the local latent variable z_i more important when decoding. Experiments show that HT-HVAE alleviates the problem of posterior collapse.

3.2. Transformer

The Transformer [24] is a new sequence-to-sequence model, different from Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), and it only relies on the attention mechanism. Since its inception, Transformer has achieved great success in many areas of natural languages, such as machine translation and other natural language understanding tasks.

The multi-head self-attention mechanism is the core of the Transformer. It mainly consists of two parts: Scaled Dot-Product Attention and Multi-Head Attention. Generally speaking, the point of Scaled Dot-Product Attention is to map a query and a key-value into an output, as shown in (1) of Figure 2. This output is the weighted sum of key values, and its weight is calculated from the similarity between query and key. In the self-attention mechanism, the similarity between query and key is scaled dot-product. Its form is as follows.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{3}$$

Q, K, and V are very common in the attention mechanism. They represent Query vector, Key vector, and Value vector, respectively, and they are all learnable parameters. By using this method, Transformer can learn the distribution of attention between words. Moreover, in order to retain a smaller inner product of Q and K, it is usually divided by the dimension of K.

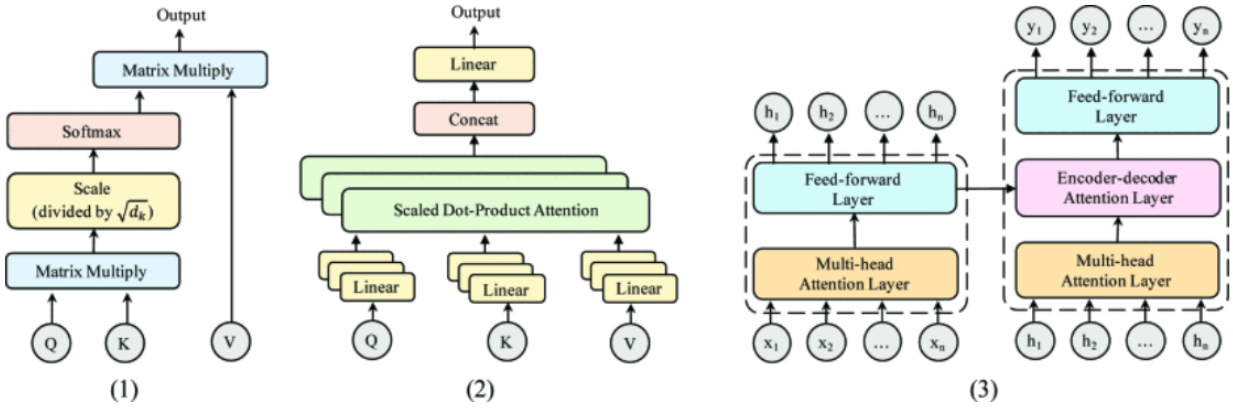

Figure 2. Transformer. (1) Scaled Dot-Product Attention, (2) Multi-Head Attention, and (3) Transformer model [24].

In addition, Transformer uses a method called multi-head self-attention mechanism, shown in the (2) of Figure 2. First, different linear functions are used to divide Q, K, and V into different h parts. Each part's self-attention score is then calculated, and these h parts are then concatenated through the linear function as the final output:

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h) \tag{4}$$

$$wherehead_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{5}$$

where the matrices $QW_i^Q \in R^{d_{model}*d_k}$, $KW_i^K \in R^{d_{model}*d_k}$, and $VW_i^V \in R^{d_{model}*d_k}$ represent the corresponding linear projections.

(3) In Figure 2, the entire structure of the Transformer is presented: The left one is the encoder, and the right one is the decoder. In particular, the Transformer's encoder consists of a multi-head self-attention layer and a feed-forward fully connected layer. Such modules have N layers in total, and each layer's module accepts the output of the previous layer as the input of the layer. The encoder's function is to encode the input x into H and then to send H to the decoder.

The decoder is similar to the encoder. The only difference is that an encoder–decoder multi-head self-attention layer is added. An encoder–decoder attention layer is directly added to the multi-head attention layer and the feed-forward fully connected layer. There are also N such modules. The encoder–decoder multi-head self-attention layer's query is the output of the previous decoder layer, and the key and value come from the output of the encoder.

3.3. HMM

The Hidden Markov Model (HMM) is a relatively classic machine learning model. It is widely used in language recognition, natural language processing, pattern recognition, and other fields.

Of course, with the current rise of deep learning, especially the popularity of neural network sequence models such as RNN and LSTM, the status of HMM has declined.

Figure 3 shows the probability graph model of HMM, where $X_{1:T} = (X_1, X_2, \ldots, X_T)$ represents an observable variable and $Y_{1:T} = (Y_1, Y_2, \ldots, Y_T)$ represents a hidden variable. The latent variables form a Markov chain, and each observable variable x_t depends on the current latent variable y_t, and its joint probability distribution is the following.

$$p(\boldsymbol{x}, \boldsymbol{y}; \theta) = \prod_{i=0}^{n} p(y_t|y_{t-1}, \theta_s) p(x_t|y_t, \theta_t) \tag{6}$$

$\boldsymbol{x}$ and $\boldsymbol{y}$, respectively, represent observable variables and hidden variables, $p(x_t|y_t, \theta_t)$ represents the output probability, $p(y_t|y_{t-1}, \theta_s)$ represents the transition probability, and θ_t and θ_s represent the parameters of both.

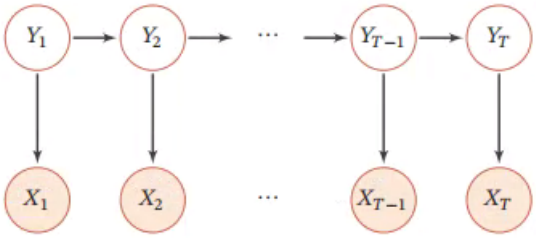

Figure 3. HMM. $X_{1:T} = (X_1, X_2, \ldots, X_T)$ represents an observable variable, and $Y_{1:T} = (Y_1, Y_2, \ldots, Y_T)$ represents a hidden variable.

4. Method

4.1. Graphical Model of HT-HVAE

Unlike the previous VAE method that only captures one latent variable, our proposed HT-HVAE captures one global latent variable (from the entire text) and multiple local latent variables (from each sentence). The global latent variable controls the local latent variables, and the following local latent variables also have a dependency on the previous local latent

variables at the same time. It is in line with the human writing habits mentioned in the Introduction of the article, that is, the semantics of the entire text or the purpose of writing controls the semantics of each sentence, and there is also a semantic conversion relationship between sentences. Figure 4 shows the Graphical Model of our method.

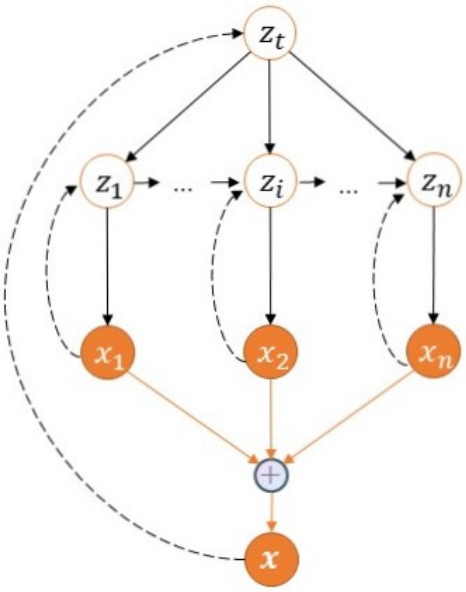

Figure 4. Graphical Model of HT-HVAE. z_t represents the global latent variable, and z_i represents the local latent variable. We assume that the global latent variable determines the local hidden variable, x represents the text data, the solid line represents the generative network, and the dashed line represents the inference network.

According to the Graphical Model, the joint probability density of the generative model is the following.

$$p(\boldsymbol{x}, \boldsymbol{z}, z_t; \boldsymbol{\theta}) = p(z_t) \prod_{i=1}^{n} p(z_i|z_{i-1}, z_t; \theta_{1i}) p(x_i|z_i; \theta_{2i}) \tag{7}$$

$\boldsymbol{x} = (x_1, \ldots, x_i, \ldots, x_n)$ and $\boldsymbol{z} = (z_1, \ldots, z_i, \ldots, z_n)$ are local latent variables, z_t represents global latent variable, and we assume that $p(z_t)$ possesses standard normal distribution; thus, it has no parameters. $\boldsymbol{\theta} = (\theta_{1i}, \theta_{2i})$ are parameters. θ_{1i} is the parameter of the prior probability distribution function of z_i, while θ_{2i} is the parameter of generative model.

Suppose that, in the inference network, z_t and $\boldsymbol{z}$ are conditionally independent given the input $\boldsymbol{x}$, and z_i is also conditionally independent given the input x_i, which is described as follows.

$$q(z_t, \boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\phi}) = q(z_t|\boldsymbol{x}; \phi_t) \prod_{i=1}^{n} q(z_i|x_i; \phi_i) \tag{8}$$

$\boldsymbol{\phi} = (\phi_t, \phi_i)$ are parameters. In the generative network, given that the local latent variable z_i is determined by the global latent variable z_t and the previous local latent variable z_{i-1}, we have the following.

$$p(z_t, \boldsymbol{z}; \boldsymbol{\theta_1}) = p(z_t) \prod_{i=1}^{n} p(z_i|z_t, z_{i-1}; \theta_{1i}) \tag{9}$$

Under the assumptions of (8) and (9), the ELBO yields the following.

$$\begin{aligned} logp(\boldsymbol{x}) &= log\int_{z_t}\int_{\boldsymbol{z}} p(\boldsymbol{x},\boldsymbol{z},z_t) = logE_{q(z_t,\boldsymbol{z}|\boldsymbol{x})}\frac{p(z_t)\prod_{i=1}^{n}p(z_i|z_{i-1},z_t)p(x_i|z_i)}{q(z_t|\boldsymbol{x})\prod_{i=1}^{n}q(z_i|x_i)} \\ &\geq E_{q(z_t,\boldsymbol{z}|\boldsymbol{x})}log\frac{p(z_t)\prod_{i=1}^{n}p(z_i|z_{i-1},z_t)p(x_i|z_i)}{q(z_t|\boldsymbol{x})\prod_{i=1}^{n}q(z_i|x_i)} \\ &= \sum_{i=1}^{n}E_{q(z_i|x_i)}logp(x_i|z_i) - D_{KL}(q(z_t,\boldsymbol{z}|x)||p(z_t,\boldsymbol{z})) \\ &= \sum_{i=1}^{n}E_{q(z_i|x_i)}logp(x_i|z_i) - E_{q(z_t|\boldsymbol{x})}log\frac{q(z_t|\boldsymbol{x})}{p(z_t)} \\ &- \sum_{i=1}^{n}E_{q(z_t,\boldsymbol{z}|\boldsymbol{x})}log\frac{q(z_i|x_i)}{p(z_i|z_{i-1},z_t)} \end{aligned} \tag{10}$$

If we create the following:

$$\begin{aligned} I &= E_{q(z_t,\boldsymbol{z}|\boldsymbol{x})}log\frac{q(z_i|x_i)}{p(z_i|z_{i-1},z_t)} = \int_{z_t}\int_{\boldsymbol{z}}q(z_t|\boldsymbol{x})\prod_{i=1}^{n}q(z_i|x_i)log\frac{q(z_i|x_i)}{p(z_i|z_{i-1},z_t)} \\ &= \int_{z_t}\int_{z_{i-1}}q(z_t|\boldsymbol{x})q(z_{i-1}|x_{i-1})q(z_i|x_i)log\frac{q(z_i|x_i)}{p(z_i|z_{i-1},z_t)} \\ &= E_{q(z_t|\boldsymbol{x})}E_{q(z_{i-1}|x_{i-1})}q(z_i|x_i)log\frac{q(z_i|x_i)}{p(z_i|z_{i-1},z_t)} \\ &= E_{q(z_t|\boldsymbol{x})}E_{q(z_{i-1}|x_{i-1})}D_{KL}(q(z_i|x_i)||p(z_i|z_{i-1},z_t)) \end{aligned} \tag{11}$$

we, thus, obtain the following.

$$\begin{aligned} L_{HT-HVAE} = ELBO &= \sum_{i=1}^{n}E_{q(z_i|x_i;\phi_i)}logp(x_i|z_i;\theta_{2i}) - D_{KL}(q(z_t|\boldsymbol{x};\phi_t)||p(z_t)) \\ &- \sum_{i=1}^{n}E_{q(z_t|\boldsymbol{x};\phi_t)}E_{q(z_{i-1}|x_{i-1};\phi_{i-1})}D_{KL}(q(z_i|x_i;\phi_i)||p(z_i|z_{i-1},z_t;\theta_{1,i-1})) \end{aligned} \tag{12}$$

We assume that both the prior distribution and the posterior distribution are in the form of Gaussian distributions; two KL divergences can then be obtained for closed-form solutions.

We deleted HMM from HT-HVAE to prove the effectiveness of HMM. The encoder and decoder are the same as those in our model, but the exclusion of HMM, which is the prior distribution of z_i, is only determined by z_t. The difference from hVAE is that each sentence will learn $q(z_i|x_i)$ and sample from it during training. The ELBO becomes the following.

$$\begin{aligned} L_{HT-VAE} = ELBO &= \sum_{i=1}^{n}E_{q(z_i|x_i;\phi_i)}logp(x_i|z_i;\theta_{2i}) - D_{KL}(q(z_t|\boldsymbol{x};\phi_t)||p(z_t)) \\ &- \sum_{i=1}^{n}E_{q(z_t|\boldsymbol{x};\phi_t)}D_{KL}(q(z_i|x_i;\phi_i)||p(z_i|z_t;\theta_{1i})) \end{aligned} \tag{13}$$

We hereby name it HT-VAE.

4.2. Model Architecture

We introduced the probability graph form of the model and derived ELBO in the previous section. This section introduces the specific architecture of the model. The Model Architecture is shown in Figure 5.

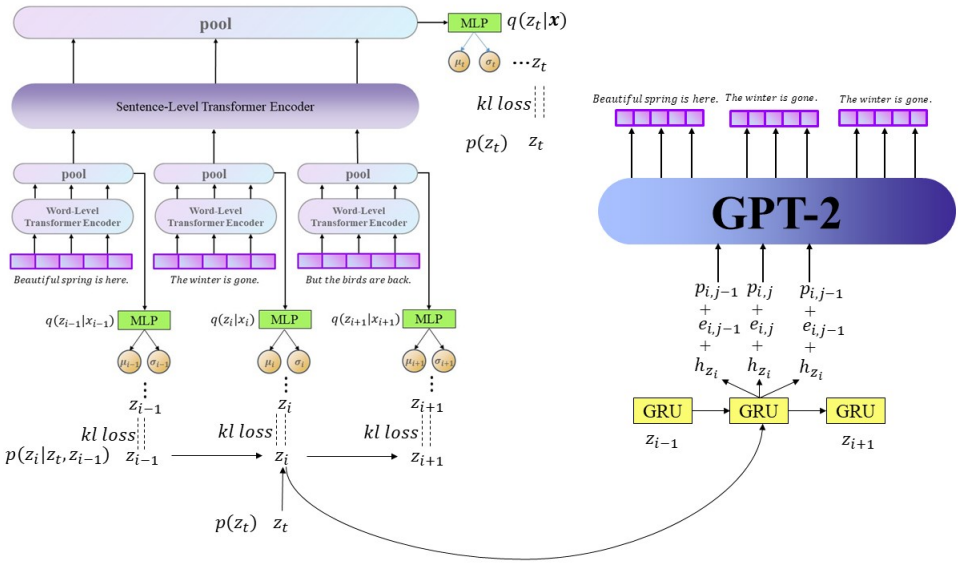

Figure 5. Model Architecture.The left side represents the inference network, and the right side represents the generative network.

4.2.1. Inference Network

The inference network is also called the encoder, and its main function is to fit $q(z|x)$ through a neural network. As mentioned previously, the HT-HVAE contains a global latent variable and multiple local latent variables; thus, we use two layers of Transformer to encode words and sentences separately and learn $q(z_i|x_i)$ and $q(z_t|\boldsymbol{x})$.

Suppose that the text x has n sentences, and each sentence has s words. We use the word-level Transformer to encode each sentence and the Transformer to encode each sentence sharing parameters. The output of the word-level Transformer is $\boldsymbol{H}^w = (H_0^w, \ldots, H_i^w, \ldots, H_s^w)$. We take H_0^w as the wave code of the sentence and input the H_0^w obtained from each sentence plus the position code into the sentence-level Transformer to obtain the code $\boldsymbol{H}^s = (H_0^s, \ldots, H_i^s, \ldots, H_n^s)$ of the entire text and take H_0^s as the text coding.

$$H_i^w = Transformer_{word}(x_i), 1 \leq i \leq n \tag{14}$$

$$H^s = Transformer_{sen}(H_i^w), 1 \leq i \leq n \tag{15}$$

The advantage of using hierarchical Transformer coding is that it can learn the attention of words and words within a sentence and the attention between sentences. For long texts, there are often semantic relations between sentences. The experiments also proved that the PPL of hierarchical coding is lower than that of non-hierarchical coding.

After obtaining the representation of each sentence H_i^w and the representation of the entire text H^s, we used a three-layer MLP network to learn the local posterior distribution and the global posterior distribution.

$$\mu_i^{'}, \sigma_i^{2'} = MLP(H_i^w) \tag{16}$$

$$\mu_t^{'}, \sigma_t^{2'} = MLP(H^s) \tag{17}$$

$\mu_i^{'}$ and $\sigma_i^{2'}$ are the mean and variance of $q(z_i|x_i)$. $\mu_t^{'}$ and $\sigma_t^{2'}$ are the mean and variance of $q(z_t|\boldsymbol{x})$. We have to learn multiple local posterior distributions, and these MLPs that learn the local posterior distribution share parameters.

4.2.2. Generative Network

In the generative network, the decoder is used to estimate $p(x|z)$. Its input is a hidden variable z, and its output is text x. In the standard VAE, z is taken from $q(z|x)$ when

training, and z is taken from the prior distribution $p(z)$ when generating text. In our model, the same is true. However, the difference is that the z_i generated by each sentence is taken from a different $q(z_i|x_i)$ during training, and z_i is taken from a different $p(z_i|z_t, z_{i-1})$ during generating.

The ELBO is mentioned in the third section, and we need to calculate KL divergences. We assume that $p(z_t)$ is the standard Gaussian distribution, $D_{KL}(q(z_t|\boldsymbol{x})||p(z_t))$ can obtain a numerical solution. However, $p(z_i|z_t, z_{i-1})$ is temporarily unknown, and another KL divergence cannot be obtained; thus, we need the MLP network to estimate $p(z_i|z_t, z_{i-1})$. z_t, and z_{i-1} is the vector that is used in the reparametrization trick to sample from $q(z_t|\boldsymbol{x})$ and $q(z_{i-1}|x_{i-1})$.

$$\mu_i, \sigma_i^2 = MLP(z_t, z_{i-1}) \tag{18}$$

μ_i and σ_i are the mean and variance of $p(z_i|z_t, z_{i-1})$. At this time, all KL terms can be found to be closed-form solutions. Next, we discuss the reconstruction item, which is also the most important part of the decoder.

We also use a hierarchical structure in the generative network. In the sentence-level decoder, we use the GRU network. The input of each GRU is z_i sampled from the posterior distribution $q(z_i|x_i)$ of each sentence, and what we call the plan vector h_i is generated. Next, h_i will be copied into s parts (s is the length of each sentence), which is used as word-level input along with word embedding and position coding.

$$h_i = GRU(z_i) \tag{19}$$

$$G_{in} = e_{ij} + h_{ij} + p_{ij} \tag{20}$$

e_{ij} represents the word embedding of the jth word of the i-th sentence in the text, and p_{ij} is the position code, $h_{ij} = h_i$.

We used 117M pretrained GPT-2 and G_{in} as the input of GPT-2 in order to obtain H_{in}. Then, it will proceed through the linear layer in order to obtain the final logits. We will add h_i and H_{in} as the input of the linear layer to further strengthen the application of hidden variables in the word-level decoder.

5. Experiment

Our experiment mainly includes unconditional long text generation and conditional long text generation. For unconditional text generation, we used the Arxiv paper abstract dataset and Yelp comment dataset; for conditional generation, we used the Arxiv paper abstract dataset. For the evaluation of the model, we mainly evaluated the language model and the generated text. Since our paper mainly studies VAE for the generation of long texts, we only consider similar models. We use hVAE [29], GPT-2 [8], and OPTIMUS [33] as the benchmarks for language models; for the evaluation of generated text, we use ARAE [51], hVAE, GPT-2, and OPTIMUS as benchmarks.

5.1. Dataset

We used the Arxiv paper abstract dataset and Yelp review dataset to verify the effectiveness of our model. Arxiv paper abstracts are more logical, while Yelp comments are not as logical as Arxiv paper abstracts.

We have performed some processing on the two datasets as we are concerned about the generation of long text: (1) First, use the GPT2 Tokinzer to segment each data text, calculate the number of characters, and select the number of characters between 150 and 300. The average number of characters in the long text is about 200 (much greater than the number of characters in hVAE and Optimus). (2) Use nltk to segment the text for the selected original text data and insert '<BOS>' at the beginning of the first sentence, insert '<EOS>' at the end of each sentence, and insert '<|endoftext|>' at the end of the last sentence. These inserted characters are not used during encoding and are instead used in the decoding stage. (3) We selected 20 million data texts as the training set and 30,000 texts as the test set. From Table 1, we can observe the specific information of the

dataset. Both datasets are divided into a training set of 200,000 long text data and a test set of 30,000 long text data. At the same time, the average number of words in Arxiv is 211, the average number of characters in Yelp is 206, and the average number of sentences in the two datasets is 6.

Table 1. Dataset. "Ave Toks" is average number of words, and Ave Sents" is the average number of sentences.

	Train	Test	Ave Toks	Ave Sents
Arxiv	200,000	30,000	211	6
Yelp	200,000	30,000	206	6

5.2. Language Model

hVAE: The encoder uses hierarchical CNN encoding, and the decoder uses a hierarchical LSTM network to learn two layers of hidden variables, but there is only one local hidden variable.

GPT-2: Composed of Transformer decoder, we use a model containing 117M parameters here.

OPTIMUS: The encoder is composed of Bert, and the decoder is composed of GPT-2.

This section mainly compares PPL and KL on the Arxiv dataset and Yelp dataset.

When evaluating the language model, we randomly selected 3000 pieces of data from the test set and chose their average PPL and KL as the final result.

However, in the VAE training process, the optimization function we chose was ELBO, which is less than or equal to $logp(x)$, and $logp(x)$ cannot be completely obtained. Therefore, PPL can only be approximated in VAE. Here, we used the IWAE method to solve OPTIMUS $logp(x)$ as accurately as possible and drew on the idea of IWAE in order to derive a method to solve the PPL of HT-HVAE with two layers of latent variables as accurately as possible:

$$\begin{aligned} logp(x) &= logE_{(z_t,\boldsymbol{z})\sim q(z_t,\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi})} \frac{1}{k} \sum_{k=1}^{n} M_k \\ &\geq \frac{1}{k} logE_{(z_t,\boldsymbol{z})\sim q(z_t,\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\phi})} \sum_{k=1}^{n} M_k = L_k \end{aligned} \tag{21}$$

where $M_k = \frac{p(\boldsymbol{x},\boldsymbol{z}_k,z_{t,k};\boldsymbol{\theta})}{q(z_{t,k},\boldsymbol{z}_k|\boldsymbol{x};\boldsymbol{\phi})}$. According to IWAE, L_k converges to $logp(x)$ as k $\to \infty$. We need to decompose $p(\boldsymbol{x},\boldsymbol{z}_k,z_{t,k};\boldsymbol{\theta})$ and $q(z_{t,k},\boldsymbol{z}_k|\boldsymbol{x};\boldsymbol{\phi})$ to obtain PPL.

$q(z_{t,k},\boldsymbol{z}_k|\boldsymbol{x};\boldsymbol{\phi})$, where z_t and z_i are independent of each other, then z_t and z_i can be sampled in $q(z_t|\boldsymbol{x})$ and $q(z_i|x_i)$.

We can observe from Table 2 that the PPL of HT-HVAE is three points lower than that of OPTIMUS, and KL divergence is also improved. If we delete HMM from our model, we will find that the KL divergence is lower than the model with HMM, and PPL is not as good. This shows that our model is better than OPTIMUS in the long text language model, which also proves that our previous assumptions are correct. That is, the latent variable information obtained from the long text as a whole cannot contain the latent variable information of each sentence, and there is a relationship between the hidden variables of each sentence. Similarly to people writing articles, there is always an outline first, and then sentences are written with one sentence after another. Finally, a long text is formed. In our model, z_t plays the role of outline, z_i is the latent variable that generates each sentence, and the relationship between sentences is expressed by HMM, which is similar to people's writing habits.

Table 2. PPL and KL.

	Arxiv		**Yelp**	
Model	KL	PPL	KL	PPL
hVAE	12.7	54.3	6.8	45.8
GPT-2	-	26.95	-	31.45
OPTIMUS	14.48	25.71	14.50	26.91
HT-VAE	18.14	24.31	19.78	26.28
HT-HVAE	**19.00**	**22.70**	**21.83**	**25.14**

We calculate the NLL loss of GPT-2, OPTIMUS, HT-VAE, and HT-HVAE on two datasets in order to further illustrate the effectiveness of HMM in decoding. It can be observed from Table 3 that the NLL loss of the HT-HVAE model is lower than that of other Transformer types, which shows that adding HMM is helpful for language modeling. In particular, it works better on the more logical Arxiv paper abstract data set.

Table 3. NLL loss.

	GPT-2	**OPTIMUS**	**HT-VAE**	**HT-HVAE**
Arxiv	695.56	686.98	668.27	**657.99**
Yelp	703.19	670.02	669.66	**667.73**

5.3. Evaluation of the Generated Text

We used BLEU [52] to evaluate the quality of the generated text and self-BLEU to evaluate the diversity of the generated text. BLEU was proposed by IBM in 2002 for the evaluation of machine translation tasks. Its overall idea is accuracy. If a reference is given, the sentence generated by the neural network is a candidate, the sentence length is n, and m words in the candidate appear in the reference. We used this method to generate 1000 sentences as candidates and used all test set data (30,000 items) as references to calculate BLEU-2, BLEU-3, and BLEU-4, respectively.

It can be observed from Table 4 that our model surpassed the original model in B-2, B-3, and B-4 scores, reaching the state of art. This shows that our model can produce better quality text. We found that in a dataset with strong logic, the BLEU score with HMM is better than the score without HMM; on the contrary, the results of the two are similar.

Table 4. BLEU.

	Yelp			**Arxiv**		
Model	B-2	B-3	B-4	B-2	B-3	B-4
hVAE	0.912	0.755	0.549	0.825	0.657	0.460
GPT-2	0.961	0.876	0.639	0.925	0.782	0.537
OPTIMUS	0.961	0.918	0.829	0.903	0.801	0.645
HT-VAE	0.972	**0.930**	**0.837**	0.911	0.809	0.651
HT-HVAE	**0.972**	0.929	0.833	**0.938**	**0.851**	**0.717**

We use self-BLEU [53] to evaluate the diversity of the generated text. The lower self-BLEU is, the better text diversity is. Similarly, we randomly generated 1000 yelp comments in order to evaluate the diversity of the text. It can be observed from Table 5 that our model may not be the best at the B-2 level but surpassed the previous model at the B-3 and B-4 levels. This shows that our model generates better text diversity.

Table 5. Self-BLEU.

	B-2	B-3	B-4
ARAE	0.725	0.544	0.402
hVAE	0.851	0.723	0.729
GPT-2	0.871	0.616	0.326
OPTIMUS	0.773	0.527	0.316
HT-VAE	0.769	0.513	0.294
HT-HVAE	0.763	**0.500**	**0.281**

The effect of our model on the yelp dataset is not much different from the baseline model, but it performs better on the Arxiv paper abstract dataset. A possible reason for this is that the corpus of Arxiv paper abstracts is richer than the Yelp dataset, and the abstract is written carefully and logically, so it is more in line with the ideas of our model. In contrast, the corpus of the Yelp comment dataset is not very rich, and the logic is definitely not as good as the Arxiv paper abstract, so our model is not much improved compared to the baseline model. The reason why our model generates better corpus diversity is due to the fact that the hidden variables that control the generation of each sentence are constantly changing, unlike other baseline models that do not change.

5.4. Interpolating Latent Space

According to reference [22], we measured the continuity of the learned latent variable space. First, we randomly sampled two points (denoted as z_{tA} and z_{tB}) from the space of the prior hidden variable z_t, and then we sent them to the decoder to generate text based on the equidistant intermediate points along the linear trajectory between z_{tA} and z_{tB}, that is, the hidden variable input into the decoder is $z_t = a * z_{tA} + (1 - a) * z_{tB}, 0 \leq a \leq 1$. As shown in Table 6, these intermediate samples are all realistic looking reviews that are syntactically and semantically reasonable, demonstrating the smoothness of the learned VAE latent space.

Table 6. Interpolating latent space. The decimal on the left represents the value of a and the generated text on the right.

0.0	My family and I went here for dinner on a Saturday night. We had a large group and were seated right away. We had a lot of fun with the food and the service. We had the "Ribs" fried rice. The rice was great! The shrimp and grits were really good and the red potatoes were nice and crispy. The wine selection was good. The atmosphere was nice and the food was great. The service was good and the food was great. We tried the "Fajita" chicken wings. I would recommend this place to anyone in the area. The wings were great and were well seasoned. They were very flavorful and the portions were huge. The bill was around $20. I was also impressed with the friendly staff. I will be back!
0.1	I'm a fan of their food and I love their beer selection. I tried their House of Beer and it's something I would have enjoyed if I could. They have a selection of beers and their beers are always pretty good. I've also tried their brunch buffet and there is a huge selection. My only problem is that they don't have a lot of seating. The staff is super nice and they are always very courteous to all the guests. If you're in the area and need to park, head over there and check this place out.
0.2	The restaurant is great. Our server was amazing. He's very friendly and polite.We've had a few reservations and it's always a good time. I also got the same server that I had last night. He's very friendly and helpful. The only reason I have not come back is because of the price. I paid $5 for a side of oysters and no oysters. We have a group of 8 people and the food is pretty good. We also ordered 2 sides and 2 appetizers. They're amazing. I'm not a huge fan of salads, but if you have salads and/or a lot of veggies, they will make it delicious. I have had more than one salad in my entire life and they are the best salad I've ever had. I'm not a fan of small portions, but if you have small portions, you will love it. I always have ordered the chicken and oyster and the rolls are delicious. I'll come back here again.

Table 6. *Cont.*

0.3	I have been here a couple of times. They are very busy and usually only have to wait a couple of hours. When I go in there is a selection of fresh, homemade, and locally made tacos. If you are craving Mexican food, I suggest the smoked steak taco (sigh). I had the goat cheese taco which is very delicious. The beef in the goat cheese was tender and juicy. I also got the chipotle chips which were nice and crunchy. The tapas were a bit dry and I'm still not sure how they were prepared. My boyfriend got the guacamole taco which was amazing.It had a really good guacamole flavor to it. It was also a little cold for me. But overall, the service was great. They have a small patio area with a large outdoor seating area. It's also convenient to eat there for a quick lunch or dinner. I would definitely come back here again and try some of their other items.
0.4	I used to love this place, but when I moved to Austin, I wasn't really sure what to expect. I started with a fish taco and a good sized selection of delicious seafood. The wait time was definitely worth it.I'd had my fish tacos before and all the people in the back were very accommodating. My friend had a plate of salmon tacos, and they were good. The tuna taco was perfect, and the fish was delicious and fresh. The chips were huge and the portions were huge. There were three different meats that I thought were excellent. The chips were super fresh and delicious. The chorizo tacos were okay, but the fish tacos were okay. The only thing I didn't like was that there was a small side of fish and that my friend didn't want to eat it. The fish tacos were pretty good too, but the price was a little too much for what I was getting. If I could give it 5 stars, I'd probably give it 5 stars.
0.5	I have been coming here for a year now. I have had a lot of bad experiences with the staff. One of the bad experiences was that the manager called the girl on the phone to tell her I was being watched. After being told that I was being watched, she ended up trying to tell me I was being watched. I was on the phone for a few minutes to talk with a manager at this time, who was not there. The manager never came back to check on me.After checking in with a manager, she was completely ignored. I had to call the manager to tell her about the problem. I have never had an issue with the staff at a McDonald's. If the staff at a McDonald's was really bad, why is the manager going to do anything to help them?
0.6	This place is just so amazing! I have been here a few times, but the first time I went it was just a small place with the only other person in the place. So the food was great, the service was fast, and the drinks were great. My friends and I were there at 7 pm on a Sunday (we were just there for a party) and were told it would be closed on the following Saturday. This made the wait even longer, and we were told it would be closed on a Saturday night. (This was done to accommodate our party's schedule). The waitress was so nice and accommodating, and the food was delicious. The service was quick, but the food was very undercooked. I am not a big fan of chicken and it was like a chicken and egg dish, but this was the best I've had.I'd recommend this place if you are looking for a great meal to spend your weekend. I'll definitely be back!
0.7	We have been to Oahu's most beautiful Hawaiian restaurants in the past, and have always had great service. The wait staff is friendly, friendly, and have a great sense of humor. I like to try their drinks. The food is delicious. We've ordered a small number of their specials and have never had a bad experience. We have also ordered several of their "merchants" and have had great customer service. I can't wait to come back to visit these other places in the area. I don't know if they are running out of good things to say or just are trying to improve the overall experience.However, I know this place will stay busy for years to come. If you love Hawaiian food, consider coming here.
0.8	This is a bar that I have been to many times in the past, and I love it. I have been to a couple bars in the area before, and this is a different bar. I had been to the Ole Miss bar before, and this time it was better. The bartenders are very nice, and they have a nice setting, which is nice. There is a decent amount of people working, and the bartenders are all friendly and polite, which is a great thing. The food is delicious. I had the shrimp appetizer, and it was soooo good! I also had the pescatarian, which was a bit over cooked, but it was really good. The main courses were the chicken, salmon and tuna, and the salad. The salmon was so good, and I thought the dressing was a bit too oily. I could see it in the picture. The salad was a little on the salty side, but it was so good.I also had the steak and onion soup, which was pretty good. The dish was a little over cooked, but it was so good, it was almost like eating an egg on top of the spinach salad.The drinks were good, too. The food is pretty good, too.But, it's just my opinion on this bar. I wouldn't go back for that, but I will give it another shot!

Table 6. *Cont.*

0.9	I wish I could give 4 stars but I have to admit I was pleasantly surprised at how well the service was. The sushi was excellent and the hot springs were great. The wait staff was great as well. The server was very friendly and had a great sense of humor. I had the sweet potato shrimp with tomato sauce and a tuna fish salad. I also had the crab cake with corn bread. The crab cake was good but was definitely a bit too small for my liking. The salmon was a bit oily and I was disappointed. I can't really say I would come back. The service was great. It was a great time. It took me about 15 min to get my drink, and after that I was able to get my drink from the counter. It was a nice little break and I didn't feel rushed by the wait staff. The drinks were also nice. The service was very good. I would definitely come back.
1.0	So I walked in and it was a little boring. I like the fact that it's a retail place. The staff was friendly and very welcoming, and I can say I had a good experience. The service was quick and friendly. The waiters are super friendly, friendly, and considerate. I had a little too much food and the bar seemed empty. I went on a Saturday night and the hostess came out and said she was going to be late, and that it was a bad night. I'm not sure why she's not in there. I had a group of 20 people come in, and the hostess said the food was pretty good, but it wasn't the best I've had. I really wanted to try this place but I ended up not being there.

5.5. Conditional Text Generation

We used the Arxiv paper abstract dataset in order to achieve controllable text generation. While encoding each word, the title of the paper is also encoded but not sent to the sentence-level encoder. At the same time, we also changed the prior probability of z_t, and it is no longer the standard Gaussian distribution but the Gaussian distribution determined by the title code c. In the generation stage, the word of the title will also be added before '<BOS>', but this part of the title will not add hidden variables unlike the sentence of the text; that is, this part of the hidden variable is zero. Given the title, the generated text is shown in Table 7.

Table 7. Conditional text generation.

Title: The Infrared Astronomical Mission AKARI
In this work we report on the IR Astronomical Mission AKARI (AKARI) based on the Z-critical stage for the optical observation of the Galactic dark energy payloads. The mission will be able to record and analyze the infrared and diffuse light from the gas clouds of the Galactic centre. MAGIC from the cosmic microwave background is anticipated to be several hundred times more powerful at deep wavelengths than the Hubble Space Telescope. We will be able to observe radiofrequency signals from the radio galaxies in the brightest regions of galaxies. We will be able to link the radio emission with the biological activity of the galaxy, thereby providing the closest scientific opportunity to examine the evolution of the Galactic Dark Energy.
Title: Polarized States and Domain Walls in Spinor Bose-Einstein Condensates
A representative dataset of spinor Bose-Einstein condensates (BSF) is investigated through a special set of independent finite fields. Switching of the field between spinor states and domain walls is considered in the sense that each spinor can be viewed as a spinor-pole. The corresponding degree of freedom is found to be in agreement with the classical relations, and the associated scaling function is modeled as a sum of two functions. It is shown that, for a given spinor, the resulting derivative of the momentum is in good agreement with the classical one. The temperature dependence, as well as a generalization of the thermodynamic stability, of the corresponding free energy are modeled in terms of spinors and their interactions. The results reveal a remarkable resemblance to experimental results, with the temperature dependence being nearly identical to those obtained by investigating spinor equivalence among spinors for finite length fields.

6. Conclusions

In this paper, we propose a new method combining VAE+HMM+Hierarchical Transformers to generate better long texts. Unlike traditional VAE, our model learns more than one hidden variable including the global hidden variable and multiple local hidden

variables. In addition, the global hidden variable and the previous local hidden variable control the current local hidden variable. Another advantage of our model is that the VAE and Transformer are further combined, and each local latent variable is sent to the decoder GPT-2 so that GPT-2 notices the connection between the local latent variables at the sentence level. This is similar to human writing habits. Experimental results prove that our model generates high-quality long texts.

However, our method still has some shortcomings. We only proposed that there are some connections between the global hidden variables and local hidden variables of the long text, which is also proved by the comparison experiment results. However, we did not conduct a quantitative analysis of the relationship between this hidden variable and analyze the strength of the relationship between them. The question of how to quantitatively analyze the relationship between such hidden variables is a difficult one to answer. This is exactly what we aim to study next.

Author Contributions: Conceptualization, K.Z.; methodology, K.Z.; validation, K.Z.; investigation, H.D. and K.Y.; writing—original draft preparation, K.Z.; writing—review and editing, K.Z.; visualization, K.Z.; supervision, X.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been supported by National Key Research and Development Program of China (NO.2018YFC1604000).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data sharing is not applicable for this article.

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

References

1. Jiang, M.; Liang, Y.; Feng, X.; Fan, X.; Pei, Z.; Xue, Y.; Guan, R. Text classification based on deep belief network and softmax regression. *Neural Comput. Appl.* **2016**, *29*, 61–70. [CrossRef]
2. Kowsari, K.; Brown, D.E.; Heidarysafa, M.; Meimandi, K.; Gerber, M.; Barnes, L.E. HDLTex: Hierarchical Deep Learning for Text Classification. In Proceedings of the 16th IEEE International Conference on Machine Learning and Applications (ICMLA), Cancun, Mexico, 18–21 December 2017; pp. 364–371.
3. Kowsari, K.; Heidarysafa, M.; Brown, D.E.; Meimandi, K.; Barnes, L.E. RMDL: Random Multimodel Deep Learning for Classification. In Proceedings of the ICISDM '18, Lakeland, FL, USA, 9–11 April 2018.
4. Lai, S.; Xu, L.; Liu, K.; Zhao, J. Recurrent Convolutional Neural Networks for Text Classification. In Proceedings of the AAAI 2015, Austin, TX, USA, 25–30 January 2015.
5. Wang, Z.; Wang, X.; An, B.; Yu, D.; Chen, C. Towards Faithful Neural Table-to-Text Generation with Content-Matching Constraints. *arXiv* **2020**, arXiv:2005.00969.
6. Zhang, Y.; Wang, G.; Li, C.; Gan, Z.; Brockett, C.; Dolan, B. POINTER: Constrained Text Generation via Insertion-based Generative Pre-training. *arXiv* **2020**, arXiv:2005.00558.
7. Dathathri, S.; Madotto, A.; Lan, J.; Hung, J.; Frank, E.; Molino, P.; Yosinski, J.; Liu, R. Plug and Play Language Models: A Simple Approach to Controlled Text Generation. *arXiv* **2020**, arXiv:1912.02164.
8. Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I. Language Models are Unsupervised Multitask Learners. 2019. Available online: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf (accessed on 2 September 2021).
9. Wang, T.; Wan, X.; Jin, H. AMR-To-Text Generation with Graph Transformer. *Trans. Assoc. Comput. Linguist.* **2020**, *8*, 19–33. [CrossRef]
10. Koncel-Kedziorski, R.; Bekal, D.; Luan, Y.; Lapata, M.; Hajishirzi, H. Text Generation from Knowledge Graphs with Graph Transformers. In Proceedings of the NAACL-HLT 2019, Minneapolis, MN, USA, 6–7 June 2019.
11. Pham, D.H.; Le, A.C. Learning multiple layers of knowledge representation for aspect based sentiment analysis. *Data Knowl. Eng.* **2018**, *114*, 26–39. [CrossRef]
12. Tao, J.; Zhou, L.; Feeney, C. I Understand What You Are Saying: Leveraging Deep Learning Techniques for Aspect Based Sentiment Analysis. In Proceedings of the HICSS 2019, Maui, HI, USA, 8–11 January 2019.
13. Liu, S.; Chen, J.H. A multi-label classification based approach for sentiment classification. *Expert Syst. Appl.* **2015**, *42*, 1083–1093. [CrossRef]

14. Chaturvedi, I.; Ong, Y.; Tsang, I.; Welsch, R.; Cambria, E. Learning word dependencies in text by means of a deep recurrent belief network. *Knowl. Based Syst.* **2016**, *108*, 144–154. [CrossRef]
15. Kim, J.; Jang, S.; Choi, S.; Park, E.L. Text Classification using Capsules. *arXiv* **2020**, arXiv:1808.03976.
16. Aly, R.; Remus, S.; Biemann, C. Hierarchical Multi-label Classification of Text with Capsule Networks. In Proceedings of the ACL 2019, Florence, Italy, 28 July–2 August 2019.
17. Ren, H.; Lu, H. Compositional coding capsule network with k-means routing for text classification. *arXiv* **2018**, arXiv:1810.09177.
18. Goodfellow, I.J.; Bengio, Y.; Courville, A.C. Deep Learning. *Nature* **2015**, *521*, 436–444.
19. Sutskever, I.; Vinyals, O.; Le, Q.V. Sequence to Sequence Learning with Neural Networks. In Proceedings of the NIPS 2014, Montreal, QC, Canada, 8–11 December 2014.
20. Kingma, D.P.; Welling, M. Auto-Encoding Variational Bayes. Available online: https://arxiv.org/abs/1312.6114 (accessed on 2 September 2021).
21. Rezende, D.J.; Mohamed, S.; Wierstra, D. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In Proceedings of the ICML 2014, Beijing, China, 21–26 June 2014.
22. Bowman, S.R.; Vilnis, L.; Vinyals, O.; Dai, A.M.; Józefowicz, R.; Bengio, S. Generating Sentences from a Continuous Space. In Proceedings of the CoNLL 2016, Berlin, Germany, 11–12 August 2016.
23. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
24. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is All you Need. *arXiv* **2017**, arXiv:1706.03762.
25. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the NAACL-HLT 2019, Minneapolis, MN, USA, 6–7 June 2019.
26. Dong, L.; Yang, N.; Wang, W.; Wei, F.; Liu, X.; Wang, Y.; Gao, J.; Zhou, M.; Hon, H. Unified Language Model Pre-training for Natural Language Understanding and Generation. *arXiv* **2019**, arXiv:1905.03197.
27. Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; Zettlemoyer, L. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. *arXiv* **2020**, arXiv:1910.13461.
28. Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; Liu, P.J. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.* **2020**, *21*, 140:1–140:67.
29. Shen, D.; Çelikyilmaz, A.; Zhang, Y.; Chen, L.; Wang, X.E.; Gao, J.; Carin, L. Towards Generating Long and Coherent Text with Multi-Level Latent Variable Models. *arXiv* **2019**, arXiv:1902.00154.
30. Wang, W.; Gan, Z.; Xu, H.; Zhang, R.; Wang, G.; Shen, D.; Chen, C.; Carin, L. Topic-Guided Variational Autoencoders for Text Generation. *arXiv* **2019**, arXiv:1903.07137.
31. Zhang, X.; Yang, Y.; Yuan, S.; Shen, D.; Carin, L. Syntax-Infused Variational Autoencoder for Text Generation. *arXiv* **2019**, arXiv:1906.02181.
32. Liu, D.; Liu, G. A Transformer-Based Variational Autoencoder for Sentence Generation. In Proceedings of the 2019 International Joint Conference on Neural Networks (IJCNN), Budapest, Hungary 14–19 July 2019; pp. 1–7.
33. Li, C.; Gao, X.; Li, Y.; Li, X.; Peng, B.; Zhang, Y.; Gao, J. Optimus: Organizing Sentences via Pre-trained Modeling of a Latent Space. *arXiv* **2020**, arXiv:2004.04092.
34. Wang, T.; Wan, X. T-CVAE: Transformer-Based Conditioned Variational Autoencoder for Story Completion. In Proceedings of the IJCAI 2019, Macao, China, 10–16 August 2019.
35. Cho, K.; Van Merrienboer, B.; Gülçehre, Ç.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv* **2014**, arXiv:1406.1078.
36. Burda, Y.; Grosse, R.B.; Salakhutdinov, R. Importance Weighted Autoencoders. Available online: https://arxiv.org/abs/1509.00519 (accessed on 2 September 2021).
37. Miao, Y.; Yu, L.; Blunsom, P. Neural Variational Inference for Text Processing. *arXiv* **2016**, arXiv:1511.06038.
38. Yang, Z.; Hu, Z.; Salakhutdinov, R.; Berg-Kirkpatrick, T. Improved Variational Autoencoders for Text Modeling using Dilated Convolutions. *arXiv* **2017**, arXiv:1702.08139.
39. Semeniuta, S.; Severyn, A.; Barth, E. A Hybrid Convolutional Variational Autoencoder for Text Generation. *arXiv* **2017**, arXiv:1702.02390.
40. Serban, I.; Sordoni, A.; Lowe, R.; Charlin, L.; Pineau, J.; Courville, A.C.; Bengio, Y. A Hierarchical Latent Variable Encoder-Decoder Model for Generating Dialogues. In Proceedings of the AAAI 2017, San Francisco, CA, USA, 4–9 February 2017.
41. Kim, Y.; Wiseman, S.; Miller, A.; Sontag, D.; Rush, A.M. Semi-Amortized Variational Autoencoders. In Proceedings of the ICML 2018, Stockholm, Sweden, 10–15 July 2018.
42. Kaiser, Ł.; Roy, A.; Vaswani, A.; Parmar, N.; Bengio, S.; Uszkoreit, J.; Shazeer, N.M. Fast Decoding in Sequence Models using Discrete Latent Variables. In Proceedings of the ICML 2018, Stockholm, Sweden, 10–15 July 2018.
43. Hashimoto, T.; Guu, K.; Oren, Y.; Liang, P. A Retrieve-and-Edit Framework for Predicting Structured Outputs. In Proceedings of the NeurIPS 2018, Montreal, QC, Canada, 3–8 December 2018.
44. Zhao, T.; Zhao, R.; Eskénazi, M. Learning Discourse-level Diversity for Neural Dialog Models using Conditional Variational Autoencoders. In Proceedings of the ACL 2017, Austin, TX, USA, 6–15 October 2017.
45. Li, B.; He, J.; Neubig, G.; Berg-Kirkpatrick, T.; Yang, Y. A Surprisingly Effective Fix for Deep Latent Variable Modeling of Text. In Proceedings of the EMNLP/IJCNLP 2019, Hong Kong, China, 3–7 November 2019.

46. Fu, H.; Li, C.; Liu, X.; Gao, J.; Çelikyilmaz, A.; Carin, L. Cyclical Annealing Schedule: A Simple Approach to Mitigating KL Vanishing. In Proceedings of the NAACL 2019, Minneapolis, MN, USA, 2–7 June 2019.
47. Higgins, I.; Matthey, L.; Pal, A.; Burgess, C.P.; Glorot, X.; Botvinick, M.; Mohamed, S.; Lerchner, A. beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. In Proceedings of the ICLR 2017, Toulon, France, 24–26 April 2017.
48. Dieng, A.B.; Kim, Y.; Rush, A.M.; Blei, D. Avoiding Latent Variable Collapse With Generative Skip Models. *arXiv* **2019**, arXiv:1807.04863.
49. He, J.; Spokoyny, D.; Neubig, G.; Berg-Kirkpatrick, T. Lagging Inference Networks and Posterior Collapse in Variational Autoencoders. *arXiv* **2019**, arXiv:1901.05534.
50. Fang, L.; Li, C.; Gao, J.; Dong, W.J.; Chen, C. Implicit Deep Latent Variable Models for Text Generation. In Proceedings of the EMNLP/IJCNLP 2019, Hong Kong, China, 3–7 November 2019.
51. Zhao, J.; Kim, Y.; Zhang, K.; Rush, A.M.; LeCun, Y. Adversarially Regularized Autoencoders. In Proceedings of the ICML 2018, Stockholm, Sweden, 10–15 July 2018.
52. Papineni, K.; Roukos, S.; Ward, T.; Zhu, W.J. Bleu: A Method for Automatic Evaluation of Machine Translation. In Proceedings of the ACL 2002, Philadelphia, PA, USA, 6–12 July 2002.
53. Zhu, Y.; Lu, S.; Zheng, L.; Guo, J.; Zhang, W.; Wang, J.; Yu, Y. Texygen: A Benchmarking Platform for Text Generation Models. In Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, Ann Arbor, MI, USA, 8–12 July 2018.