Hindawi
Mathematical Problems in Engineering
Volume 2018, Article ID 7529286, 8 pages
https://doi.org/10.1155/2018/7529286

Research Article

An Integrated Deep Generative Model for Text Classification and Generation

Zheng Wang and Qingbiao Wu

School of Mathematical Sciences, Zhejiang University, HangZhou, Zhejiang, China

Correspondence should be addressed to Zheng Wang; twflf.wz@163.com

Received 10 June 2018; Accepted 7 August 2018; Published 19 August 2018

Academic Editor: Mustansar A. Ghazanfar

Text classification and generation are two important tasks in the field of natural language processing. In this paper, we deal with both tasks via Variational Autoencoder, which is a powerful deep generative model. The self-attention mechanism is introduced to the encoder. The modified encoder extracts the global feature of the input text to produce the hidden code, and we train a neural network classifier based on the hidden code to perform the classification. On the other hand, the label of the text is fed into the decoder explicitly to enhance the categorization information, which could help with text generation. The experiments have shown that our model could achieve competitive classification results and the generated text is realistic. Thus the proposed integrated deep generative model could be an alternative for both tasks.

1. Introduction

Text classification is one of the most basic and important tasks in the field of natural language processing, in which one should assign predefined categories to the text. The result of classification is often used as the input for other tasks; thus an efficient and accurate classification algorithm is of great benefit. Traditional classification models combined with traditional representation of the text such as support vector machine base on the bag-of-word vectors or other classical methods have been able to achieve good results in some simple application scenarios [1–4].

In recent years, deep learning models based on neural networks have achieved remarkable results in various tasks, such as computer vision [5] and speech recognition [6]. Obviously, the above models have also played a significant role in the field of natural language processing. A considerable part of the task is based on the word vector representations, which are learned through neural language models [7–10]. These word vector representations, also known as word embeddings, are transformed by a deep neural network such as convolutional neural network (CNN) or recurrent neural network (RNN) to obtain more abstract features about the text [11–14]. Thus we could perform the text classification based on these features. Such methods often lead to more advanced results on larger dataset[15].

On the other hand, text generation is also an important task of concern. Many works about neural language models or sequence to sequence models have been made to improve the quality of the generated text for a variety of purposes [16–18]. Compared with other models, deep generative models have stronger expressive power, have the potential to handle more types of data, and have the natural advantage of being able to generate sample from the models, indicating their potential to deal with the above task.

Variational Autoencoder (VAE), which is a powerful deep generative model, has attracted the attention of many researchers in recent years [19, 20]. It consists of a probability encoder and a probability decoder and takes advantages of the variational inference. A variational lower bound is optimized instead of the traditional log-likelihood.

Variational Autoencoder has proved its ability, both in theory and in practice, thus could be chosen to perform the text generation task. What is more, the encoder of the VAE could be regarded as a feature extractor, which could be utilized to perform the classification task. In this paper, we propose an integrated model based on VAE to handle both generation and classification tasks. Although VAE has

shown impressive advances in visual domain, such as image classification and image generation, its application in natural language processing has been relatively less studied [21]. In our work, we use a modified version of VAE to extract global features about the text, while using the text label to facilitate the generation of text during the decoding stage. The influence of the label information is enhanced by explicitly feeding label to the decoder at each time step. For text classification, we train a classifier based on neural network and integrate it with the VAE. Experiments have shown that our model can handle text classification and text generation problems at the same time and could achieve ideal results.

The remainder of the paper is structured as follows. In Section 2 we review the important architecture of VAE, which is the foundation of our method and experiments. In Section 3 we introduce the basic RNN structure. In Section 4 we elaborate the details of the proposed method. In Section 5 we introduce some related works. We provide the experimental evaluation and show the results in Section 6. Finally we make a conclusion in Section 7.

2. Review of VAE

In this section, we review the basic VAE model as the foundation of our work.

Variational methods provide an optimization-based alternative to the sampling-based Monte Carlo methods in learning the complex latent variables models. Variational methods try to approximate the true posterior distribution by minimizing the Kullback-Leibler divergence between the true posterior distribution and a simple probability distribution which may be predefined and tractable. For instance, mean-field variational method [22] approximate the true posterior distribution with a fully factorized set of distributions. Recently, some stochastic variational inference methods have been proposed to update the variational parameters directly by sampling from the variational posterior distribution [23–25].

Variational Autoencoder (VAE)[19, 26] has proved to be one of the most successful probabilistic generative models, which combines the variational learning framework and deep neural network. It could be regarded as a generative model that is based on a regularized version of the standard autoencoder.

In VAE, the deterministic encoder $\varphi_e nc$ is replaced by a learned posterior recognition model $q_\phi(\mathbf{z} \mid \mathbf{x})$. Similarly, there is a probabilistic decoder $p_\theta(\mathbf{x} \mid \mathbf{z})$.

The objective function is the same as the Evidence Lower Bound (ELBO) in variational learning, which takes the following form:

$$\begin{aligned}\log p_\theta(\mathbf{x}) &= \log \sum_{\mathbf{z}} p_\theta(\mathbf{x}, \mathbf{z}) = \log \sum_{\mathbf{z}} q_\phi(\mathbf{z} \mid \mathbf{x}) \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z} \mid \mathbf{x})} \\ &\geq \sum_{\mathbf{z}} q_\phi(\mathbf{z} \mid \mathbf{x}) \log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z} \mid \mathbf{x})} \\ &= \sum_{\mathbf{z}} q_\phi(\mathbf{z} \mid \mathbf{x}) \log p_\theta(\mathbf{x}, \mathbf{z}) + H(q) = ELBO\end{aligned} \tag{1}$$

where H(·)is the entropy.

The ELBO could be reformulated as

$$-D_{KL}\left(q_\phi(\mathbf{z} \mid \mathbf{x}) \parallel p_\theta(\mathbf{x})\right) + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x} \mid \mathbf{z})\right] \tag{2}$$

In practice, the prior over the latent variables is usually chosen to be the centered isotropic multivariate Gaussian $p_\theta(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})$. The variational approximate posterior is chosen to be a multivariate Gaussian with a diagonal covariance structure whose distribution parameters are computed from $\mathbf{z}$ with a fully connected neural network with a single hidden layer:

$$\log q_\phi\left(\mathbf{z} \mid \mathbf{x}^{(i)}\right) = \log \mathcal{N}\left(\mathbf{z}; \boldsymbol{\mu}^{(i)}, \boldsymbol{\sigma}^{2(i)}\mathbf{I}\right) \tag{3}$$

where $\boldsymbol{\mu}^{(i)}$ and $\boldsymbol{\sigma}^{(i)}$ are outputs of the encoding MLP.

In the traditional stochastic variational learning framework, the Evidence Lower Bound is optimized with respect to both the generative parameters θ and the variational parameters ϕ. The gradient with respect to θ could be calculated by

$$\nabla_\theta \mathscr{L}(x) = E_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\nabla_\theta \log p_\theta(\mathbf{x}, \mathbf{z})\right] \tag{4}$$

$$\nabla_\theta \mathscr{L}(x) \approx \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta \log p_\theta\left(\mathbf{x}, \mathbf{z}^{(i)}\right) \tag{5}$$

while the gradient with respect to ϕ could be calculated by

$$\begin{aligned}\nabla_\phi \mathscr{L}(x) &= E_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\nabla_\theta \log p_\theta(\mathbf{x}, \mathbf{z}) - \log q_\phi(\mathbf{z} \mid \mathbf{x})\right] \\ &= E_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\left(\log p_\theta(\mathbf{x}, \mathbf{z}) - \log q_\phi(\mathbf{z} \mid \mathbf{x})\right)\nabla_\phi \right. \\ &\quad \left. \cdot \log q_\phi(\mathbf{z} \mid \mathbf{x})\right]\end{aligned} \tag{6}$$

$$\begin{aligned}\nabla_\phi \mathscr{L}(x) \approx \frac{1}{n}\sum_{i=1}^{n} &\left(\log p_\theta\left(\mathbf{x}, \mathbf{z}^{(i)}\right) - \log q_\phi\left(\mathbf{z}^{(i)} \mid \mathbf{x}\right)\right) \\ &\times \nabla_\phi \log q_\phi\left(\mathbf{z}^{(i)} \mid \mathbf{x}\right)\end{aligned} \tag{7}$$

Unfortunately, this gradient estimator exhibits high variance, and some other methods should be used to alleviate it.

In VAE, the so-called reparameterization trick is used to solve the above problem. Assume that $\mathbf{z}$ is a continuous random variable and could be sampled by $\mathbf{z} \sim q_\phi(\mathbf{z} \mid \mathbf{x})$. Then $\mathbf{z}$ sometimes could be expressed as a deterministic variable $\mathbf{z} = g_\phi(\boldsymbol{\epsilon}, \mathbf{x})$, where $\boldsymbol{\epsilon}$ is an auxiliary variable with independent marginal $p(\boldsymbol{\epsilon})$ and $g_\phi(\cdot)$ is some vector valued function parameterized by ϕ.

Given $\mathbf{z} = g_\phi(\boldsymbol{\epsilon}, \mathbf{x})$, we have

$$q_\phi(\mathbf{z} \mid \mathbf{x}) \prod_i dz_i = p(\boldsymbol{\epsilon}) \prod_i d\epsilon_i \tag{8}$$

$$\begin{aligned}\int q_\phi(\mathbf{z} \mid \mathbf{x}) f(\mathbf{z}) d\mathbf{z} &= \int p(\boldsymbol{\epsilon}) f(\mathbf{z}) d\boldsymbol{\epsilon} \\ &= \int p(\boldsymbol{\epsilon}) f\left(g_\phi(\boldsymbol{\epsilon}, \mathbf{x})\right) d\boldsymbol{\epsilon};\end{aligned} \tag{9}$$

thus we could construct the estimator by

$$\int q_\phi(\mathbf{z} \mid \mathbf{x}) f(\mathbf{z}) d\mathbf{z} \simeq \frac{1}{L}\sum_{l=1}^{L} f\left(g_\phi\left(\boldsymbol{\epsilon}^{(l)}, \mathbf{x}\right)\right) \tag{10}$$

where $\epsilon^{(l)} \sim p(\boldsymbol{\epsilon})$.

For instance, assume that $z \sim p(z \mid x) = \mathcal{N}(\mu, \sigma^2)$. Then z could be reparameterized by $z = \mu + \sigma\epsilon$, where $\epsilon \sim \mathcal{N}(0,1)$. We have

$$\begin{aligned}\mathbb{E}_{\mathcal{N}(z;\mu,\sigma^2)}[f(z)] &= \mathbb{E}_{\mathcal{N}(\epsilon;0,1)}[f(\mu+\sigma\epsilon)] \\ &\simeq \frac{1}{L}\sum_{l=1}^{L} f\left(\mu + \sigma\epsilon^{(l)}\right)\end{aligned} \tag{11}$$

where $\epsilon^{(l)} \sim \mathcal{N}(0,1)$

3. Recurrent Neural Network

Neural networks have been applied to a variety of tasks in the field of natural language processing. In particular, recurrent neural networks with long short-term memory (LSTM) [27, 28] cells or gated recurrent units (GRU)[29] have proven successful at tasks including machine translation [16–18], machine comprehension, and many others. These models are especially suitable for handling sequential data.

Due to the vanishing and exploding gradient problems, it is widely believed that learning long-range dependencies with recurrent neural networks is challenging. To deal with this issue, LSTM introduces a more complex interaction structure. The recurrent computations in LSTM can be represented by

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \tag{12}$$

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \tag{13}$$

$$\widetilde{C}_t = \tanh\left(W_C \cdot [h_{t-1}, x_t] + b_C\right) \tag{14}$$

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C}_t \tag{15}$$

$$o_t = \sigma\left(W_o \cdot [h_{t-1}, x_t] + b_o\right) \tag{16}$$

$$h_t = o_t * \tanh(C_t) \tag{17}$$

where x_t is the input, h_t is the cell's state, C_t is the cell's memory, $\widetilde{C}_t$ is the cell's candidate memory, f_t, i_t, o_t are the state of the corresponding gate, and $W_f, W_i, W_C, W_o, b_f, b_i, b_C, b_o$ are parameters of the cell.

It should be noted that there are many variants of LSTM, but throughout this paper, we always use the standard LSTM.

4. Details of the Model

In this section, we introduce the structure of the model in detail.

4.1. The Probabilistic Encoder. In VAE, the encoder would encode the input into the hidden code. To handle text data, we use LSTM as the main part of the encoder. Given a sequence consisting of N words $\{x_1, x_2, \ldots, x_N\}$, we first convert each word into an embedding vector $w_i \in \mathbb{R}^{d_{emb}}$. The embeddings are encoded by column vectors in an embedding matrix $W_{emb} \in \mathbb{R}^{d_{emb} \times |V|}$, where d_{emb} is the dimension of the embedding and $|V|$ is the size of the vocabulary. Each column corresponds to the embedding of the i-th word in the vocabulary. The matrix W_{emb} is a parameter to be learned and the dimension of the embedding is a hyperparameter to be chosen by the user.

The embeddings of the words are then fed into the LSTM cell step by step, and finally we get the hidden state $h_1, h_2, \ldots, h_N$ of all the N timesteps. Unlike the vanilla sequential VAE[21], here we prefer to use bidirectional LSTM and introduce the self-attention mechanism to more efficiently extract the information from the entire text. The variational posterior represented by the encoder would be a multivariate Gaussian with a diagonal covariance structure, where the mean and the standard deviation of the posterior are set to be the outputs of the MLP. Formally, we have

$$\mathbf{H}_{fw} = LSTM_{enc}(x_1, x_2, \ldots, x_N; \phi_{enc}) \tag{18}$$

$$\mathbf{H}_{bw} = LSTM_{enc}(x_N, x_N - 1, \ldots, x_1; \phi_{enc}) \tag{19}$$

$$\mathbf{H}_{full} = \left[\mathbf{H}_{fw}; \mathbf{H}_{bw}\right] \tag{20}$$

$$\mathbf{a} = softmax\left(w_{s2} \tanh\left(W_{s1}\mathbf{H}_{full}^T\right)\right) \tag{21}$$

$$h_C = \mathbf{a}^T \mathbf{H}_{full} \tag{22}$$

$$\boldsymbol{\mu} = sigmoid(W_0 h_C + b_0) \tag{23}$$

$$\boldsymbol{\sigma} = sigmoid(W_1 h_C + b_1) \tag{24}$$

$$q(\mathbf{z} \mid \mathbf{x}) = \mathcal{N}\left(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2\mathbf{I}\right) \tag{25}$$

where $\phi_{enc}, W_{s1}, w_{s2}, W_0, W_1, b_0, b_1$ are parameters to be learned. In the above procedure, $\mathbf{H}_{fw}, \mathbf{H}_{bw} \in \mathbb{R}^{N \times h_{dim}}$ represent all the hidden state of the forward and backward directions of LSTM. $\mathbf{H}_{full} \in \mathbb{R}^{N \times 2h_{dim}}$ is the concatenation of $\mathbf{H}_{fw}$ and $\mathbf{H}_{bw}$. $\mathbf{a} \in \mathbb{R}^N$ is the weight vector used in self-attention mechanism. $h_C \in \mathbb{R}^{2h_{dim}}$ is the weighted average of all the hidden state, also called the context state vector.

4.2. The Text Classifier. Since text classification is one of our aims, we can use the hidden code z as the extracted abstract feature of text. Ideally, we assume that the hidden code contains important information about the input text, such as semantics, length, and sentiment. Based on these features, we can use traditional classifiers, such as SVM and AdaBoost to classify texts. It is important to note that the categorization of text is only one of our goals and our other goal is the generation of text, while the class information about the text helps the latter. Therefore, we consider integrating the classifier into the VAE model and train the neural network as classifier.

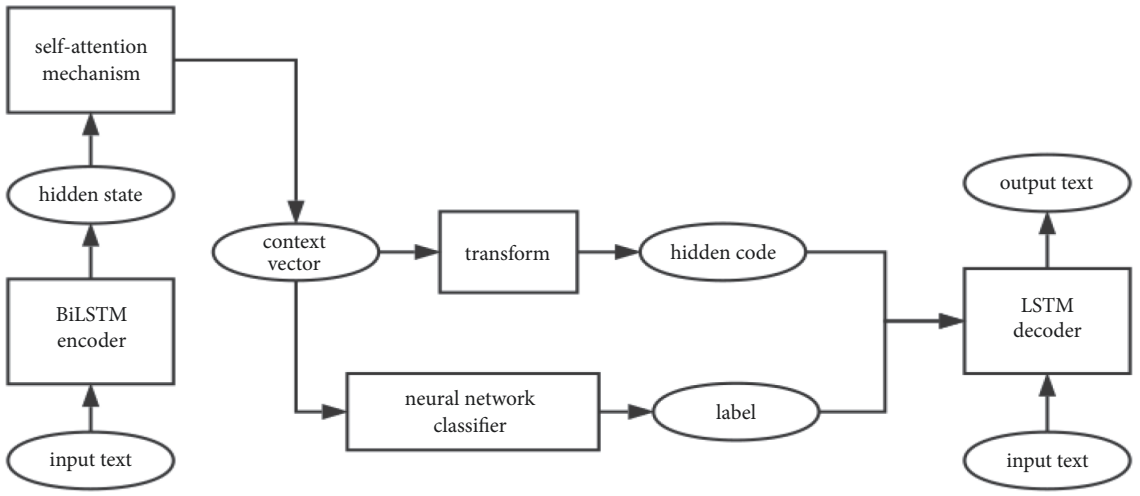

Figure 1: The framework of the proposed model.

The hidden code is fed into a fully connected layer followed by a softmax layer, and the output is a vector indicating the probability of each class. Formally, we have

$$q_c(\mathbf{y}) = softmax(W_c\mathbf{z} + b_c) \tag{26}$$

where W_c, b_c are parameters to be learned.

4.3. The Probabilistic Decoder. The decoder would generate the word sequentially, conditioned on the information provided by the encoder. We again use LSTM as the main part of the decoder. The procedure of word embedding remains unchanged. Decoding of timestep t could be expressed as

$$h_t = LSTM_{dec}(x_1, x_2, \ldots, x_{t-1}, z; \theta_{dec}) \tag{27}$$

$$p(x_t) = softmax(W_o h_t + b_o) \tag{28}$$

$$\widehat{x_t} \sim p(x_t) \tag{29}$$

where θ_{dec}, W_o, b_o are parameters to be learned.

Recall that we have the categorization information for the text. Thus we may use the label to help controlling the generation progress. A natural idea is to condition on both the hidden code and the label at the beginning of the decoding procedure. Unfortunately, this often does not achieve good performance in practice. As the sequence gets longer, the signal provided by the class label gets weaker rapidly. To make full use of the label, we concatenate on the word embedding and the one-hot label vector at each timestep, and the hidden state of each timestep could be represented by

$$h_t = LSTM_{dec}(x_1, x_2, \ldots, x_{t-1}, z, y; \theta_{dec}) \tag{30}$$

The remaining steps are the same as before.

4.4. The Objective Function. As our model needs to take into account both text classification and generation, the objective function we need to optimize also consists of two parts. Given a labeled data pair $(\mathbf{x}, y)$, we have

$$p_\theta(\mathbf{x} \mid \mathbf{z}, y) = \prod_t p_\theta(x_t \mid \mathbf{x}_{<t}, \mathbf{z}, y) \tag{31}$$

and the Evidence Lower Bound could be expressed as

$$\begin{aligned} \log p_\theta(x) \geq\ & \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z, y)\right] \\ & - D_{KL}\left(q_\phi(z \mid x) \parallel p_\theta(z)\right) = -\mathscr{L}_g(x) \end{aligned} \tag{32}$$

where $D_{KL}(q \parallel p)$ is the Kullback-Leibler divergence between the distribution q and p.

The loss of the classifier could be expressed as

$$\mathscr{L}_c(x, y) = -\sum_y p_e(y) \log q_c(y \mid x) \tag{33}$$

where $p_e(y)$ represents the empirical distribution of the ground truth label.

The final objective function for the entire dataset is

$$\mathscr{L}_{total} = \sum_{(x,y)} \left(\mathscr{L}_g(x) + \alpha \mathscr{L}_c(x, y)\right) \tag{34}$$

where α is the hyperparameter controlling the trade-off between these two parts.

The reparameterization trick and sampling method could be used to calculate the gradient with respect to each parameter. Then the optimization would be applied.

The framework of our model is shown in Figure 1.

5. Related Works

In recent years, VAE has proved to be a powerful deep generative model. The first work to combine VAE with text generation is [21]. They use the vanilla LSTM as both the encoder and the decoder. They introduced some tricks for training VAE such as word drop and KL annealing, which

Table 1: Details about the used datasets.

Dataset	number of classes	training	validation	testing
IMDB	2	25000	10000	15000
SST1	5	8544	1101	2210
SST2	2	6920	872	1821

we also used in our work. But their work do not include the classification task. At the same time, their model cannot generate text for a given category by controlling the label.

There are some similar works on the same topic such as [36, 37]. In [36], they use the vanilla LSTM as the encoder of VAE, and the hidden code of the VAE is transformed from the hidden state of the last timestep in the LSTM. This means that the LSTM should compress the information into the last hidden state as much as possible, which is hard due to the sequential structure of the vanilla LSTM. In fact, the information of the previous timesteps is often lost. In contrast, our model use bidirectional LSTM as the encoder, which could capture more contextual information than the vanilla one. More importantly, we introduced the self-attention mechanisms [38, 39]; the final context state vector would be derived from all the hidden state in the LSTM rather than just the last hidden state. The extracted feature would contain more global information and could be used to improve the performance of both the classification and generation tasks. What is more, although they use a modified version of CNN-based decoder, they do not explicitly take advantage of the label during decoding stage, and the categorical signal may not be strong enough to guide the generation procedure, while our method explicitly uses the label information during decoding and it would help controlling the generation progress. In [37], they also use the vanilla LSTM as encoder, which would be less robust and less efficient.

6. Experiments

In this section we show the experimental results on several datasets to demonstrate the performance of the proposed method. First we would introduce the dataset we used in the experiments; then we would describe the details of the experiments; finally we would show the results of the experiment from two perspectives, text classification and text generation.

6.1. Datasets. We performed the model on three different benchmarks: IMDB dataset, SST1, and SST2.

The IMDB dataset is a benchmark for sentiment classification [30]. The task is to determine if the movie reviews are positive or negative. The Stanford Sentiment Treebank (SST) dataset consists of movie reviews with one sentence per review [35]. In SST1, the fine-grained labels are provided, including very positive, positive, neutral, negative, and very negative. The SST2 dataset is the same as SST1 but with neutral reviews removed and binary labels. Details about the dataset are shown in Table 1.

6.2. Experimental Setup. For all the experiments, the vocabulary size is set to 15000; the word out of the dictionary is replaced by the $< unk >$ token. In practice, it is unlikely to handle sentences of arbitrary length; thus we would set the max length of the sentence to be 35. We would truncate the sentence if the length exceeds the maximum, and we would pad the sentence with the $< pad >$ token to make the length of sentences in each minibatch consistent. During the decoding procedure, we would add the $< bos >$ and $< eos >$ token at the beginning and end of the sentence, respectively.

Table 2: The classification error rate (%) on the IMDB dataset of each method. Lower is better.

Methods	Error rate on IMDB
SVM + bow(baseline)	11.55
WRRBM+bow[30]	10.77
SVM+bow2+NB[31]	8.78
NB-LM + bow3[32]	8.13
LSTM	13.50
VAE+NN(ours)	8.86

We use the pretrained GloVe vectors [40] with 300 dimensions as the initialization of the word embedding. During training, these embeddings are optimized as part of the parameters of the model. The hidden size of the LSTM cell is set to 128. The size of the hidden code z is also set to 128. The batch size is set to 64. The classifier is a neural network with two hidden layers, and the size of each layer is set to 200 and 400.

We implemented the model with Tensorflow. The model is trained end-to-end using the ADAM optimizer, with learning rate of $4e - 3$.

In order to better train the VAE model, the cost annealing trick is adopted to smooth the training by gradually increasing the weight of the KL divergence from zero to one. With this trick, we would avoid vanishingly small KL term in the VAE module. We also use word dropout to regularize the model. In other words, the tokens fed into the decoder would be replaced by the $< unk >$ token with a certain probability. We set this ratio to 0.25 during training, while in the decoding stage, this trick would not be used.

All the other hyperparameters, such as the trade-off between the two terms in the cost function, are chosen based on the performance on the development set.

6.3. Classification Performance. To evaluate the proposed model, we first train a SVM classifier based on the bag of words feature as a baseline model on each dataset. These models are implemented by LIBSVM. We also compared our model with a few previous best methods.

The classification results on IMDB dataset are shown in Table 2. Some previous best supervised results are contained.

Table 3: The classification error rate (%) on the SST dataset of each method. Lower is better.

Methods	Error rate on SST1	Error rate on SST2
SVM + bow(baseline)	61.4	22.8
RAE[33]	56.8	17.6
MV-RNN[34]	55.6	17.1
RNTN[35]	54.3	14.6
DCNN[13]	51.5	12.2
VAE+NN(ours)	55.3	16.9

For instance, the NB-LM bow3 method first generates binary bag of n-gram vectors, multiplies the component for each n-gram with the NB weight, and then trains a logistic regression classifier.

We found that although our model does not outperform some best methods, the proposed model could still get a competitive result, which indicates that the feature extracted by the encoder is meaningful. We emphasize that the best models are specially optimized for classification task, while our model also needs to consider the task of text generation. Therefore, simply using the result of classification as the basis of judgement does not fully demonstrate the ability of our model.

We also carried out a series of similar experiments on the SST dataset. The results are shown on Table 3. This time we compared some best methods based on neural networks, such as the Recursive Autoencoders (RAE) with pretrained word vectors from Wikipedia, Matrix-Vector Recursive Neural Network (MV-RNN) with parse trees, Recursive Neural Tensor Network (RNTN) with tensor based feature function and parse trees, and Dynamic Convolutional Neural Network (DCNN) with k-max pooling. Once again, we found that our model far outperforms the baseline and could achieve competitive results.

6.4. Generation Performance. We then conducted additional experiments to demonstrate the ability of our model on text generation. In order to make full use of the label information to generate the corresponding category of text, we explicitly feed the label into the decoder. In other words, the word embedding and the one-hot label vector are concatenated at each timestep as the input.

We used the model trained on the IMDB dataset to generate text corresponding to different categories. The results are shown on Table 4. We found that the generated text is realistic while corresponding the correct label, that is, including different positive and negative emotions.

To make the experiments more complete, we also evaluated the performance of text generation on SST dataset. The samples are shown on Table 5. We found that the generated text is shorter and simpler compared with the text generated from the IMDB dataset. This is because the IMDB dataset is larger and our model would learn more complex structures. The results have demonstrated that our model could consistently achieve good generative performance.

Table 4: Samples of the text learned from the IMDB dataset.

Positive samples:
Lynch has a similar sense of humor in the movie
The music score itself is somewhat touching memorable
He left a deep impression on us and more elements would describe the movie
This movie did get serious as well and had some really emotional moments as well
It was entertaining and I still had great fun in watching it
Negative samples:
Imagination is not enough
Honestly those fans would not like to see some aspects of movies
It didn't really stick out that much
The CG is not advanced enough to make the animated dogs real
This is only a disturbing, superficial entertainment

Table 5: Samples of the text learned from the SST dataset.

Positive samples:
Emotional movie
Theatrical foreign love
The film is the very good story
She was definitely a unique film
He does a wonderful performance as well
Negative samples:
This movie is not worth the fare
It is very disappointing
Empowerment life into a depression era
There are many meaningless scenes
There was too much of what we have seen before

7. Conclusion

In this paper, we have proposed an integrated deep generative model to deal with both the text classification and generation. We use bidirectional LSTM and introduce the self-attention mechanism to enhance the encoder of VAE. We then extract the feature of the text and use the neural network classifier to perform the classification based on the global feature. What is more, the categorization information is explicitly fed into the decoder to help controlling the generation progress. The experiments have shown that our model could achieve competitive results on both tasks. While our work

has enhanced the encoder of VAE for better performance, we have not modified the main structure of the decoder. In fact, some new mechanisms can be introduced to enhance the decoder, such as coverage probability. What is more, some constraint functions could be introduced to produce more controllable text after modifying the objective function. We leave them as our future work.

Data Availability

All the datasets used in this paper are publicly available and could be obtained from http://ai.stanford.edu/~amaas/data/sentiment/ and http://nlp.stanford.edu/sentiment/.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant nos. 11771393 and 11632015) and Zhejiang Natural Science Foundation (Grant no. LZ14A010002).

References

[1] K. Vandana and C. Namrata Mahender, "Text classification and classifiers: A survey," *International Journal of Artificial Intelligence and Applications*, vol. 3, no. 2, p. 85, 2012.

[2] S. Shang, M. Shi, W. Shang, and Z. Hong, "Improved feature weight algorithm and its application to text classification," *Mathematical Problems in Engineering*, vol. 2016, Article ID 7819626, 12 pages, 2016.

[3] L. La, Q. Guo, D. Yang, and Q. Cao, "Multiclass boosting with adaptive group-based knn and its application in text categorization," *Mathematical Problems in Engineering*, vol. 2012, Article ID 793490, 24 pages, 2012.

[4] C.-M. Tan, Y.-F. Wang, and C.-D. Lee, "The use of bigrams to enhance text categorization," *Information Processing & Management*, vol. 38, no. 4, pp. 529–546, 2002.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 26th Annual Conference on Neural Information Processing Systems (NIPS '12)*, pp. 1097–1105, Lake Tahoe, Nev, USA, December 2012.

[6] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proceedings of the 38th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '13)*, pp. 6645–6649, May 2013.

[7] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of Machine Learning Research*, vol. 3, pp. 1137–1155, 2003.

[8] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of the 27th Annual Conference on Neural Information Processing Systems (NIPS '13)*, pp. 3111–3119, December 2013.

[9] K. Erk, "Vector space models of word meaning and phrase meaning: a survey," *Linguistics and Language Compass*, vol. 6, no. 10, pp. 635–653, 2012.

[10] D. Clarke, "A context-theoretic framework for compositionality in distributional semantics," *Computational Linguistics*, vol. 38, no. 1, pp. 41–71, 2012.

[11] X. Zhang, J. Zhao, and Y. Lecun, "Character-level convolutional networks for text classification," *Advances in Neural Information Processing Systems*, pp. 649–657, 2015.

[12] C. Daniel, Y. Yinfei, K. Sheng-yi et al., "Universal sentence encoder," 2018.

[13] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 655–665, Baltimore, Md, USA, June 2014.

[14] C. N. Dos Santos and M. Gatti, "Deep convolutional neural networks for sentiment analysis of short texts," in *Proceedings of the 25th International Conference on Computational Linguistics (COLING '14)*, pp. 69–78, 2014.

[15] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," in *Proceedings of the European Chapter of the Association for Computational Linguistics*, pp. 427–431, Valencia, Spain, 2017.

[16] B. Dzmitry, C. Kyunghyun, and B. Yoshua, "Neural machine translation by jointly learning to align and translate," in *Proceedings of the International Conference on Learning Representations*, San Diego, USA, 2015.

[17] T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1412–1421, Lisbon, Portugal, September 2015.

[18] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Advances in Neural Information Processing Systems*, pp. 3104–3112, 2014.

[19] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *Proceedings of the International Conference on Learning Representations*, Banff, Canada, 2014.

[20] D. P. Kingma, D. J. Rezende, S. Mohamed, and M. Welling, "Semi-supervised learning with deep generative models," in *Proceedings of the 28th Annual Conference on Neural Information Processing Systems 2014, NIPS 2014*, pp. 3581–3589, Canada, December 2014.

[21] S. R. Bowman, L. Vilnis, O. Vinyals, A. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space," in *Proceedings of the Computational Natural Language Learning*, pp. 10–21, Berlin, Germany, 2016.

[22] L. K. Saul, T. Jaakkola, and M. I. Jordan, "Mean field theory for sigmoid belief networks," *Journal of Artificial Intelligence Research*, vol. 4, no. 1, pp. 61–76, 1996.

[23] M. Andriy and K. Gregor, "Neural variational inference and learning in belief networks," in *Proceedings of the International Conference on Machine Learning*, pp. 1791–1799, Beijing, China, 2014.

[24] J. Paisley, B. David, and M. Jordan, "Variational bayesian inference with stochastic search," in *Proceedings of the International Conference on Machine Learning*, pp. 1363–1370, Edinburgh, Scotland, 2012.

[25] R. Ranganath, S. Gerrish, and D. M. Blei, "Black box variational inference," *Journal of Machine Learning Research*, vol. 33, pp. 814–822, 2014.

[26] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative

models," in *Proceedings of the International Conference on Machine Learning*, pp. 1278–1286, Beijing, China, 2014.

[27] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[28] F. A. Gers and J. Schmidhuber, "LSTM recurrent networks learn simple context-free and context-sensitive languages," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 12, no. 6, pp. 1333–1340, 2001.

[29] C. Gulcehre, K. Cho, R. Pascanu, and Y. Bengio, "Learned-norm pooling for deep feedforward and recurrent neural networks," in *Machine Learning and Knowledge Discovery in Databases*, Lecture Notes in Computer Science, pp. 530–546, Springer, Heidelberg, Berlin, Germany, 2014.

[30] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, ACL-HLT 2011*, pp. 142–150, USA, June 2011.

[31] S. Wang and C. D. Manning, "Baselines and bigrams: Simple, good sentiment and topic classification," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, ACL 2012*, pp. 90–94, Republic of Korea, July 2012.

[32] M. Grégoire, T. Mikolov, R. MarcAurelio, and Y. Bengio, "Ensemble of generative and discriminative techniques for sentiment analysis of movie reviews," 2014.

[33] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning, "Semi-supervised recursive autoencoders for predicting sentiment distributions," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '11)*, pp. 151–161, Association for Computational Linguistics, July 2011.

[34] R. Socher, B. Huval, C. D. Manning, and A. Y. Ng, "Semantic compositionality through recursive matrix-vector spaces," in *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL 2012*, pp. 1201–1211, Association for Computational Linguistics, Republic of Korea, July 2012.

[35] R. Socher, A. Perelygin, J. Y. Wu et al., "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP '13)*, pp. 1631–1642, Citeseer, October 2013.

[36] Y. Zichao, H. Zhiting, S. Ruslan, and B. Taylor, "Improved variational autoencoders for text modeling using dilated convolutions," in *Proceedings of the International Conference on Machine Learning*, pp. 3881–3890, Sydney, Australia, 2017.

[37] W. Xu, H. Sun, C. Deng, and Y. Tan, "Variational autoencoder for semi-supervised text classification," *AAAI*, pp. 3358–3364, 2017.

[38] A. Vaswani, N. Shazeer, N. Parmar et al., "Attention is all you need," *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

[39] L. Zhouhan, F. Minwei, N. D. S. Cicero et al., "A structured self-attentive sentence embedding," in *Proceedings of the International Conference on Learning Representations*, Toulon, France, 2017.

[40] J. Pennington, R. Socher, and C. D. Manning, "GloVe: global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014*, pp. 1532–1543, October 2014.