Pre-train and Plug-in: Flexible Conditional Text Generation with Variational Auto-Encoders

Yu Duan[1*], **Canwen Xu**[2*], **Jiaxin Pei**[3*], **Jialong Han**[4†], **Chenliang Li**[2‡]
[1] Alibaba Group, China [2] Wuhan University, China
[3] University of Michigan, United States [4] Amazon, United States
[1] derrick.dy@alibaba-inc.com, [2] {xucanwen,cllee}@whu.edu.cn
[3] pedropei@umich.edu, [4] jialonghan@gmail.com

Abstract

Conditional Text Generation has drawn much attention as a topic of Natural Language Generation (NLG) which provides the possibility for humans to control the properties of generated contents. Current conditional generation models cannot handle emerging conditions due to their joint end-to-end learning fashion. When a new condition added, these techniques require full retraining. In this paper, we present a new framework named **P**re-train and **P**lug-in **V**ariational **A**uto-**E**ncoder (PPVAE) towards flexible conditional text generation. PPVAE decouples the text generation module from the condition representation module to allow "one-to-many" conditional generation. When a fresh condition emerges, only a lightweight network needs to be trained and works as a plug-in for PPVAE, which is efficient and desirable for real-world applications. Extensive experiments demonstrate the superiority of PPVAE against the existing alternatives with better conditionality and diversity but less training effort.[1]

1 Introduction

Currently, neural generation techniques have powered many inspiring applications, e.g., poem generation (Yang et al., 2018), neural machine translation (NMT) (Bahdanau et al., 2015) and chatbot (Zhao et al., 2017). Conditional (also known as controllable) text generation is an important task of text generation, aiming to generate realistic text that carries a specific attribute (e.g., positive or negative sentiment). A common solution is to encode the condition into a vector representation and then integrate it with the text generation process (Kingma et al., 2014; Hu et al., 2017; Mirza and Osindero, 2014). These existing neural models have achieved encouraging results. However, when a new condition is added (e.g., a new topic for categorical generation), they require a full retraining or fine-tuning. This process is both time-consuming and computationally inefficient (Houlsby et al., 2019). Both fine-tuning and retraining are not desirable in real-world applications since the delivery (e.g., transmitting updated weights through the Internet) and client-side re-deployment (e.g., distribute updated weights to users) of large-scale weights are often difficult.

Inspired by the recent success of Variational Auto-Encoder (VAE) (Kingma and Welling, 2014) based post-hoc conditional image generation strategy (Engel et al., 2018), we provide a new perspective for flexible conditional text generation. We propose **P**re-train and **P**lug-in **V**ariational **A**uto-**E**ncoder (PPVAE), which decouples the text generation module from the condition representation module. PPVAE is a hierarchical framework composed of two VAEs: **(1) PRETRAINVAE**, which derives the global latent space of text with its encoder (pre-trained global encoder) and learns to generate text based on an easily-accessible large unlabeled dataset with its decoder (pre-trained global decoder); **(2) PLUGINVAE**, which is a lightweight neural network that learns to transform vectors from the conditional latent space to the global latent space, and vice versa. This mapping function can be easily learned with only a few conditional training samples. In this sense, once we transform a latent variable (also known as latent code) randomly sampled from the conditional space distribution to the global space, the pre-trained global decoder is directly adopted for generation. In other words, whenever a new condition emerges, we only need to train a PLUGINVAE and directly plug it into the framework.

[*] The first three authors contribute equally to this paper.
[†] Work done when Jialong Han was with Tencent AI Lab.
[‡] Chenliang Li is the corresponding author.
[1] The code is available at https://github.com/WHUIR/PPVAE.

Different from the existing end-to-end neural models (Mirza and Osindero, 2014; Sohn et al., 2015; Kingma et al., 2014), PPVAE focuses on the learning of pure transformation between the continuous latent spaces, instead of the tricky discrete text generation. Once trained, PRETRAINVAE is fixed for text representation and generation under all conditions. Our proposed framework decouples the conditional space learning from the text generation, endowing PPVAE with more flexibility when handling emerging conditions. Also, training only a small conditional network for latent space transformation is much more efficient than co-training with the text generation. Additionally, we can easily increase the capability of generation using a larger corpus or deeper neural networks for text encoding and decoding. Our main contributions can be summarized as follows: (1) We propose a novel framework, PPVAE, for conditional text generation, which allows a separate training for a new condition without retraining the whole network. (2) We conduct extensive experiments and analysis to verify the effectiveness of our proposed PPVAE. Our framework achieves state-of-the-art performance on conditionality in both automatic and human evaluations.

2 Related work

Boosted by the recent success of deep learning technology, Natural Language Generation (NLG) has recently become popular in the NLP community. Many great works have attempted to solve various subtasks like dialogue generation (Li et al., 2016), poetry generation (Yi et al., 2018) and story generation (Fan et al., 2018) and new techniques keep emerging (Bowman et al., 2016; Yu et al., 2017; Zhou et al., 2020). However, due to the black-box nature of neural networks, the recent proposed generic models suffer the problem of lacking interpretability and controllability.

To handle this problem and support generating plausible text with a specified condition, conditional text generation (Kikuchi et al., 2016; Ficler and Goldberg, 2017; Hu et al., 2017) has recently attracted extensive attention. Current research in this direction mainly falls into two fashions: the supervised methods and semi-supervised methods. For supervised methods, Mirza and Osindero (2014); Sohn et al. (2015) first converted the condition information to one-hot vectors, then integrated them into a generator and a discriminator. To enhance the correlation between structured conditional code and generated samples, Chen et al. (2016) adopted an extra adversarial classifier to infer the structured code from generated samples. Wang and Wan (2018) used multiple generators for multiple conditions and a multi-class classifier to provide training signals for the learning of generators.

However, given only a limited number of conditional samples, semi-supervised methods are compulsory. To utilize the implicit conditional distribution behind the unlabeled text, Kingma et al. (2014) introduced a classifier into the VAE architecture. Hu et al. (2017) further involved two additional independent regularization terms in enhancing the disentanglement between structured code and unstructured code. Very recently, Keskar et al. (2019) used human-defined "control code" to pre-trained Language Model in an unsupervised manner.

Our work falls in the category of semi-supervised learning yet differs from the existing works in the following ways: (1) Our model decouples the text generation module from the condition representation module which two are tightly fused as a single one in previous studies, enabling possible exploitation for pre-trained Language Models (e.g., GPT-2 (Radford et al., 2019)). (2) Our model allows single-condition generation, which could inspire new applications like polite speech generator (Niu and Bansal, 2018) and data augmentation (Guo et al., 2018). (3) Our model can handle emerging conditions while achieving state-of-the-art performance with fewer parameters and less training time.

3 Preliminaries

Variational Auto-Encoder (VAE). VAE (Kingma and Ba, 2015) is widely used in continuous generation (e.g., image generation). Bowman et al. (2016) introduced VAE to NLG to solve the "one-to-many" generation problem (i.e., generating multiple feasible samples for the same input). Given a latent variable z randomly sampled from a prior distribution, VAE comprises an encoder $enc(x) = q_\phi(z|x)$ and a decoder $dec(z) = p_\theta(x|z)$. The encoder aims to encode input data x into latent space $Z \in \mathbb{R}^d$. The decoder is used to reconstruct the original input x, given the corresponding z. Thus, the loss function of VAE is formulated as:

$$\begin{aligned}\mathcal{L}_{VAE}(x) = & -\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] \\ & + \mathrm{KL}(q_\phi(z|x) \| p(z))\end{aligned} \quad (1)$$

where $\mathrm{KL}(\cdot||\cdot)$ is the Kullback-Leibler (KL) divergence, $p(z) = \mathcal{N}(0,1)$ is the prior distribution. The first term ensures that VAE can distill compact variable z in latent space for reconstruction. The second term pushes posterior distribution to be close to the prior distribution, securing the mutual information between original data and the latent space (Dupont, 2018).

Conditional Text Generation with VAE. Conditional text generation has drawn much attention recently. By controlling the properties of generated contents, we can apply the generative models to many real-world scenarios. We follow the problem setting in (Hu et al., 2017). Given a set of k conditions $C = \{c_1, c_2, ..., c_k\}$, an unlabeled corpus X, and conditional text samples $Y = Y_1 \cup Y_2 \cup ... \cup Y_k$ where each Y_i is a set of text samples that carries the condition c_i. The goal of a VAE model is to learn a decoder $p_\theta(\hat{y}|z, c_i)$ that takes the latent variable z and the condition c_i to calculate the distribution over the text samples Y_i. Thus, when the condition c_i and a randomly sampled latent variable $z \sim p(z)$ specified, the model could generate realistic text samples matching the given condition.

4 Pre-train and Plug-in Variational Auto-Encoder

As a basis for semi-supervised learning, a large unlabeled corpus should include diverse text which covers a vast spectrum of conditions. Thus, text under each condition forms a conditional latent space, which could be mapped from a larger global latent space. Based on this, we propose a PRETRAINVAE and a PLUGINVAE to derive the global and conditional latent space, respectively.

4.1 Framework

PRETRAINVAE is composed of a pre-trained global encoder for text representation and a pre-trained global decoder for text generation.

PRETRAINVAE. The encoder and decoder of PRETRAINVAE are used to encode and generate text, respectively. As discussed above, PRETRAINVAE is trained on a large amount of unlabeled text to derive the global latent space Z_g for the latent variable z_g, where $Z_g \in \mathbb{R}^{d_g}$ and d_g is the space dimension. Previous studies usually use a common VAE for text representation and generation. However, as pointed out in (Bowman et al., 2016), VAE suffers the notorious "posterior collapse" problem. To address this, we utilize Wasserstein Autoencoder (WAE) (Tolstikhin et al., 2018) for PRETRAINVAE. Different from the original VAE, WAE encourages aggregated posterior distribution to be close to the prior, which is effective in alleviating the reconstruction problem of VAE (Tolstikhin et al., 2018). Specifically, we adopt WAE-GAN, a variant of WAE, which incorporates the merits of adversarial learning. During training, the encoder $enc_g(x) = q_g(z_g|x)$ encodes the text to the latent space and the decoder $dec_g(z_g) = p_g(x|z_g)$ reconstruct the text with the latent variable z_g. Thus, the loss function of PRETRAINVAE is formulated as:

$$\begin{aligned}\mathcal{L}_{\text{PRETRAINVAE}}(x) = &- \mathbb{E}_{q_g(z_g|x)}[\log p_g(x|z_g)] \\ &+ \lambda D(Q(z_g), p(z_g))\end{aligned} \tag{2}$$

where $Q(z_g) = \int q_g(z_g|x)p(x)\ dx$ is the aggregated posterior distribution; $p(z_g)$ is the prior normal distribution; D is the adversarial discriminator; λ is the coefficient hyper-parameter ($\lambda > 0$).

PLUGINVAE. For each condition, we use a condition-specific PLUGINVAE to derive the conditional space. That is, PLUGINVAE is proposed to learn the transformation between the conditional and global latent space for each condition. Specifically, for each condition c_i, we use a limited number of conditional samples y_i and utilize the global encoder enc_g to encode them into v_{y_i}. Note that normally, the encoded text samples under a single condition are not likely to densely clustered in the global text space Z_g, since the learning process of Z_g is condition-independent and the unlabeled corpus contains diverse text samples. PLUGINVAE for condition c_i consists of an encoder $enc_{c_i}(v_{y_i}) = q_{c_i}(z_{c_i}|v_{y_i})$ and a decoder $dec_{c_i}(z_{c_i}) = p_{c_i}(v_{y_i}|z_{c_i})$. The learned condition-dependent latent space is $Z_{c_i} \in \mathbb{R}^{d_c}$, where d_c is the space dimension. Thus, PLUGINVAE is capable of mapping the samples in the global latent space to and from a denser conditional latent space (i.e., $d_c < d_g$). During training, the loss function of PLUGINVAE for a single condition is defined as:

$$\begin{aligned}\mathcal{L}_{single}(v_{y_i}) = &-\mathbb{E}_{q(z_{c_i}|v_{y_i})}[\log p_{c_i}(v_{y_i}|z_{c_i})] \\ &+ \mid (\mathrm{KL}(q_{c_i}(z_{c_i}|v_{y_i}) \| p(z_{c_i})) - \beta \mid\end{aligned} \tag{3}$$

where $p(z_{c_i})$ is the prior normal distribution of the conditional latent space; z_{c_i} is the latent variable; $v_{y_i} = enc_g(y_i)$ is encoded text samples from Y_i. To enhance the diversity of generated text, we introduce an extra constant term β to control the amount

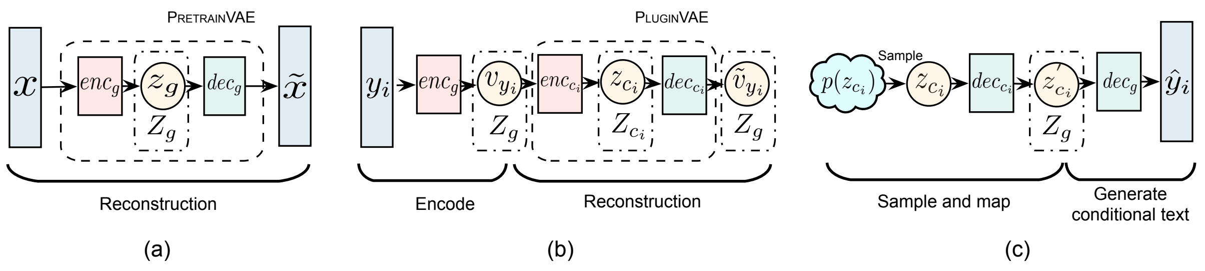

Figure 1: The whole workflow of our proposed framework.

of encoded information in VAE (Dupont, 2018; Chen et al., 2018; Kim and Mnih, 2018). By setting β to an appropriate value, PLUGINVAE could extract compact conditional information without sacrificing the fluency or accuracy.

Although we can already generate conditional text under a single condition by Equation 3, it is possible to even further improve the conditionality by introducing negative samples. We construct the negative samples $y_i^{'}$ from $Y_i^{'}$ and encode them:

$$\begin{aligned} Y_i^{'} &= Y - Y_i \\ v_{y_i}^{'} &= enc_g(y_i^{'}) \end{aligned} \tag{4}$$

Thus, the loss function of PLUGINVAE with negative samples is defined as:

$$\begin{aligned} &\mathcal{L}_{\text{PLUGINVAE}}(v_{y_i}, v_{y_i}^{'}) \\ &= \mathcal{L}_{single}(v_{y_i}) - \gamma\, \mathcal{L}_{single}(v_{y_i}^{'}) \end{aligned} \tag{5}$$

where v_{y_i} is a batch of encoded samples under condition c_i, and $v_{y_i}^{'}$ is a batch of encoded negative samples; γ is a hyper-parameter balancing the positive and negative samples. For different tasks, the best setting for γ may vary. Intuitively, the larger the difference between the conditions is, the smaller γ should be.

4.2 Workflow

In this section, we provide the details of training and generation procedures. As illustrated in Figure 1, the workflow is composed of three steps.

Pre-train once, infer everywhere. First, as shown in Figure 1(a), using the unlabeled corpus X, we pre-train PRETRAINVAE to learn the global latent space Z_g by reconstruction with Equation 2. Once pre-trained, the weights of both enc_g and dec_g are fixed. As an unsupervised VAE model, PRETRAINVAE is capable of generating diverse but unconditional text.

Train it when you need it. Previous methods (Kingma et al., 2014; Hu et al., 2017) learn the joint conditional space by jointly considering all conditions. However, once the model is trained, it is not possible to add a new condition without a full retraining. Different from those approaches, PPVAE is totally flexible that allows adding new conditions. Shown in Figure 1(b), once a condition is added, we only need to train a PLUGINVAE specifically for this condition with Equation 3 (or Equation 5, if provided with samples of other conditions). Since PLUGINVAE is text-irrelevant and only learns to map between two latent spaces, the training number of parameters is only 0.34% (see Section 6.3) of fine-tuning PRETRAINVAE or retraining other models. Additionally, although we need to train k PLUGINVAE for k conditions, the total number of trained parameters is still much smaller than existing methods (unless $k > 1/0.34\% \approx 294$, which is impossible in actual applications). Plus, we can parallel the conditional training to speed up the process easily.

Plug it in and generate. Shown in Figure 1(c), once PLUGINVAE for the condition c_i is trained, we can plug it into the PPVAE framework and generate text together with PRETRAINVAE.

First, we randomly sample a latent variable z_{c_i} from the prior distribution $p(z_{c_i}) = \mathcal{N}(0, 1)$. Then we use PLUGINVAE's decoder dec_{c_i} to map z_{c_i} to the global latent space Z_g and obtain $z_{c_i}^{'}$:

$$z_{c_i}^{'} = dec_{c_i}(z_{c_i}). \tag{6}$$

Since $z_{c_i}^{'} \in Z_g$, we can directly use the global decoder dec_g to generate text:

$$\hat{y}_i = dec_g(z_{c_i}^{'}) \tag{7}$$

where $\hat{y}_i$ is the generated text under condition c_i.

5 Experimental Settings

5.1 Datasets

Following the setting of (Hu et al., 2017), we mainly focus on short text generation (no longer

Dataset	#Train	#Dev	#Test	Avg-len
Yelp	444,101	63,483	126,670	8.93
News Titles	249,043	23,949	20,000	9.85

Table 1: The statistics of Yelp and News Titles.

than 15 tokens), which is easier for both automatic and human evaluations. We use Yelp (Shen et al., 2017) and News Titles (Fu et al., 2018) for experiments. Yelp is a collection of restaurant reviews. We use the pre-processed version used in (Shen et al., 2017), where two polarity sentiment labels are provided. For News Titles, we choose the titles belong to *Business*, *Entertainment* and *Health* categories for our experiments.

Both Yelp and News Titles are datasets with relatively short text. We filter out text longer than 15 words, then choose the top 8,900 and 10,000 words as the vocabulary for Yelp and News Titles, respectively. The statistics of the two datasets are listed in Table 1. We discard the labels in the original training and validation splits. We use the original training split as the unlabeled corpus; the validation split to select the best unsupervised models, and the test split as the labeled conditional text.

Based on the Yelp dataset, we define two tasks: (1) **Sentiment.** This task aims at generating text samples, either positive or negative. The ratio of positive/negative text in Yelp is roughly $0.6 : 0.4$. We randomly sample 200 positive and 200 negative text for supervised training. (2) **Length.** This task aims at generating text samples with a specific length. We define ($len \leq 3$) as short text, ($len \geq 12$) as long text and ($3 < len < 12$) as medium text. We respectively sample 200 text for short, medium, and long text for supervised training.

Based on the News Titles dataset, we define the categorical text generation task called **Topic.** This task aims at generating text samples on a certain topic. The ratio of business/health/entertainment in News Title is $0.38 : 0.15 : 0.47$, which is more imbalanced than Yelp. We randomly sample 200 text for each category for supervised learning.

5.2 Baselines

We use two semi-supervised methods, S-VAE (Kingma et al., 2014) and CTRL-GEN (Hu et al., 2017) as our baselines. S-VAE incorporates a classifier to provide conditional distribution for unlabeled data. Note that S-VAE is originally proposed for image generation but adapted to text generation as a baseline by Hu et al. (2017). CTRL-GEN further exploits several regularization terms to enhance the disentanglement between the structured code and the unstructured code. For a fair comparison, both the text encoder and decoder of the two baselines are the same as PRETRAINVAE. Furthermore, the baseline methods also exploit the same unlabeled corpus X and labeled corpus Y as described in the original papers.

5.3 Models

PPVAE is a model-agnostic approach, which means that both the encoders and encoders of PRETRAINVAE and PLUGINVAE can be modified to work under different settings. Here, we describe the model architecture used in our experiments.

PRETRAINVAE. For the encoder, we use a one-layer Bidirectional Gated Recurrent Unit (Bi-GRU) with 256 hidden units in each direction as its encoder. Two linear Fully-Connected (FC) layers are used for re-parameteristic trick (Kingma and Welling, 2014). For the decoder, we use a Transformer (Vaswani et al., 2017) (3 layers, 8 heads). Additionally, we add extra positional embedding after each block, and the linearly transformed encoded vector is provided as input for each block (Brock et al., 2019). For a fair comparison, we use the same encoder-decoder architecture for both S-VAE and CTRL-GEN.

PLUGINVAE. The encoder is a two-layer FC network of 64/32 hidden units taking input in d_g dimensions with an additional linear output layer of d_c units. The decoder is a two-layer FC network of 32/64 hidden units taking the latent variable in d_c dimensions as input with a linear output layer of d_g units. The activation function used in the FC networks is LeakyRelu (Maas et al., 2013).

5.4 Hyper-Parameters

PRETRAINVAE. The size of latent space d_g is set to 128. The word embedding is in 256 dimensions and randomly initialized. The output softmax matrix is tied with the embedding layer. For the adversarial classifier, we adopt two 128D hidden FC layers with LeakyRelu activation and one 1D output linear layer without bias. The balance coefficient λ is 20 for Yelp and 15 for News Titles. We train the WAE-GAN with Wasserstein Divergence (Wu et al., 2018) to smooth the training process. The coefficient k and power p of Wasserstein Divergence

Task	Conditions	Method	Accuracy (↑ better)	Log-Variance (↓ better)	Distinct-1 (↑ better)	Distinct-2 (↑ better)
Sentiment	{Positive, Negative}	S-VAE	0.7194	-5.38	0.0198	0.2520
		CTRL-GEN	0.6998	-2.78	0.0026	0.0164
		PPVAE-single *(ours)*	<u>0.7832</u>	<u>-11.12</u>	<u>0.0350</u>	<u>0.2568</u>
		PPVAE *(ours)*	**0.8484**	**-11.90**	**0.0356**	**0.2627**
Length	{Short, Medium, Long}	S-VAE	0.8598	-4.82	0.0187	0.1795
		CTRL-GEN	0.3957	-1.96	0.0021	0.0146
		PPVAE-single *(ours)*	<u>0.9640</u>	<u>-6.96</u>	**0.0375**	**0.2549**
		PPVAE *(ours)*	**0.9722**	**-7.64**	<u>0.0372</u>	<u>0.2538</u>
Topic	{Business, Health, Entmt.}	S-VAE	0.6930	-2.32	0.0360	0.2162
		CTRL-GEN	0.5335	-3.39	0.0107	0.0431
		PPVAE-single *(ours)*	<u>0.7725</u>	**-3.82**	**0.0497**	**0.3152**
		PPVAE *(ours)*	**0.8024**	<u>-3.68</u>	<u>0.0478</u>	<u>0.3056</u>

Table 2: The results of conditional text generation tasks. We use **boldface** and <u>underline</u> to indicate the best and the second-best performance. PPVAE-single indicates PPVAE with a PLUGINVAE trained under the single condition setting, as described in Section 5.5. We show the natural logarithm (ln) of variance, since the original scale is too small for demonstration.

are set to 2 and 6, respectively. During pre-training, the batch size is set to 512. Adam (Kingma and Ba, 2015) with $beta_1 = 0$ is used as the optimizer. The learning rate is set to 5×10^{-4}.

PLUGINVAE. We set the size of latent space $d_c = 20$. γ is set to 0.1 for sentiment tasks, 0.05 for categorical tasks, and 3×10^{-3} for length tasks. The batch size is set to 128. Adam (Kingma and Ba, 2015) with $beta_1 = 0.5$ is used as the optimizer, learning rate is 3×10^{-4} for 20K iterations. β linearly increases from 0 to 5 in first 10K iterations.

5.5 Evaluation Settings

Metrics. We evaluate the results with two metrics, accuracy and diversity. For *accuracy*, we train a sentiment classifier and categorical classifier (Kim, 2014), which could achieve accuracy of 90% and 97% on the validation set, respectively. The accuracy of *length* task can be directly calculated with the word count of generated text. Plus, a model that performs well on only one condition but poorly on others is not practically useful. Thus, to measure the robustness among conditions, we calculate the variance of accuracy under all conditions in a task. For *diversity*, we adopt *Distinct-1* and *Distinct-2* (Li et al., 2016) metrics. Distinct-1/Distinct-2 are the ratios of unique 1-gram/2-gram, respectively. A higher value indicates better diversity. For all tasks and models, we randomly generate 10K text for each condition by greedy decoding and report the averaged results.

Single Condition Generation. In a real-world scenario, the full set of conditions is not always available. When provided only a labeled set of target text (i.e., $k = 1$), it is not possible to learn the joint conditional space for S-VAE and CTRL-GEN any more. However, PPVAE can deal with that by training *without* negative samples using Equation 3.

6 Experimental Results

6.1 Overall Comparisons

Accuracy. The results of conditional text generation are listed in Table 2. On *sentiment* task, our model outperforms CTRL-GEN and S-VAE by 0.1486 and 0.129, respectively. On *length* task, the accuracy of our model exceeds 95%, dramatically outperforming S-VAE and CTRL-GEN by 0.1124 and 0.5765 on accuracy. Notably, the performance of CTRL-GEN (0.3957) is extremely low, demonstrating the limitation of its generator-discriminator (Goodfellow et al., 2014) training process and its token-based discriminator, which is unable to discriminate text with different lengths. On *topic* task, our model scores higher on accuracy than S-VAE and CTRL-GEN by 0.1094 and 0.2689, respectively. On all three tasks, PPVAE-single performs slightly poorer than PPVAE with negative samples, verifying the effectiveness of negative sampling. Furthermore, our models achieve the lowest variance on all three tasks, indicating that PPVAE is robust and achieves a good balance among conditions.

Diversity. Diversity is a long-lasting issue lying in the field of generative models. Recent works (Wang et al., 2017; Razavi et al., 2019) reveal the capability of the diverse content generation with

Task	Method	Fluency	Conditionality
Sentiment	S-VAE	3.10	3.04
	CTRL-GEN	**3.65**	3.23
	PPVAE-single	3.54	3.23
	PPVAE	3.30	**3.29**
Length	S-VAE	**3.64**	0.8598
	CTRL-GEN	2.53	0.3597
	PPVAE-single	3.43	0.9640
	PPVAE	3.50	**0.9722**
Topic	S-VAE	3.31	2.78
	CTRL-GEN	3.09	2.51
	PPVAE-single	3.38	3.33
	PPVAE	**3.45**	**3.57**

Table 3: Human evaluation results. Note that since the *length* task is objectively defined, we copy the accuracy results from Table 2.

VAE-based methods. These works also conclude that VAE-based methods have better output diversity than GAN-based models. Our experimental results support this conclusion well. Particularly, CTRL-GEN suffers poor diversity, which indicates the generation of "dull text" (Li et al., 2016). Both S-VAE and PPVAE show prominently better diversity than GAN-based model, CTRL-GEN. Note that the relation between the usage of negative examples and text diversity of PPVAE is not statistically prominent ($p > 0.05$).

6.2 Human Evaluation

We conduct human annotations as a complementary evaluation beyond automatic metrics. Specifically, eight individual judges are asked to rate over 200 conditional samples generated from each model and each condition. That is, for each model, a total of $4,800$ text samples are annotated. A judge needs to rate *fluency* and *conditionality* in the standard 1 to 5 scale. *Fluency* measures whether the text samples are natural and fluent as real (i.e., human-written) ones. *Conditionality* indicates whether the generated text adheres to the given condition. Shown in Table 3, PPVAE achieves the best conditionality in both automatic and human evaluations on all three tasks. Meanwhile, PPVAE retains a satisfying fluency on *sentiment* and *length* tasks and obtains the best fluency on the *topic* task.

6.3 Training Costs

To measure the efficiency of proposed methods, we report the training time and the number of parameters of S-VAE, CTRL-GEN and PPVAE in Table 4. We train the models on a single Nvidia GTX 1080 GPU and report the training time until the convergence of each model. PRETRAINVAE has the same size of S-VAE but only needs to be trained once and does not require a full retraining when a new condition added. Also, PLUGINVAE, which learns to transform between the global latent space and the conditional latent space, only has 22K parameters and can be trained within about one minute.

Method	# Training Params	Training Time
S-VAE	6.5M	1.4h
CTRL-GEN	8.5M	3.5h
PRETRAINVAE	6.5M	1.2h (only once)
PLUGINVAE	22K	64s

Table 4: Average numbers of parameters and time costs for training.

Task	Method	Acc.	Distinct-1/2
Sentiment	Fine-tuning	0.5319	0.0281 / **0.2845**
	PPVAE-single	0.7832	0.0350 / 0.2568
	PPVAE	**0.8484**	**0.0356** / 0.2627
Length	Fine-tuning	0.9456	0.0340 / **0.2923**
	PPVAE-single	0.9640	**0.0375** / 0.2549
	PPVAE	**0.9722**	0.0372 / 0.2538

Table 5: The comparisons of fine-tuned PRETRAIN-VAE with the full PPVAE on the two tasks of Yelp dataset.

β	Accuracy	Distinct-1	Distinct-2
0.0	1.0000	0.0001	0.0001
2.0	**0.9938**	0.0256	0.1629
5.0	0.9908	0.0301	0.2112
10.0	0.9875	**0.0324**	**0.2370**

Table 6: The impact of different β on long text generation task.

6.4 PLUGINVAE vs. Fine-Tuning

As a natural baseline, the conditional generation can also be done by directly fine-tuning PRETRAIN-VAE on each condition. Shown in Table 5, despite the fact that it is not computationally efficient and saving the full weights is undesirable for industrial applications when the model is large (e.g., GPT-2 (Radford et al., 2019)), both PLUGINVAE trained with and without negative samples significantly outperform a directly fine-tuned PRETRAINVAE on accuracy.

Task	Condition	Generated Examples
Sentiment	Positive	The services are friendly, fast.
	Negative	The egg drop soup was old and tasted like feet.
Length	Short	Great pricing!
	Medium	I refused to work with you and this place.
	Long	And this made me feel uncomfortable and the prices aren't right.
Topic	Business	FDA Approves New Case of E-cigarettes
	Health	Ebola : Virus Spreads in the US
	Entertainment	Surprise Birthday: The Guys of the Cast of Disney Parks

Table 7: Some conditional examples generated by PPVAE for qualitative analysis (cherry-picked).

Generated Examples
S-VAE
Chinese State Media: 17 Miners Trapped Underground
Huge Increases in Obamacare Premiums Are Coming
Herbalife Ltd. (HLF) Probe Earns Bill Ackman Back Millions
CTRL-GEN
Pfizer's Astrazeneca's Astrazeneca Bid for Astrazeneca
FDA's New Drug to Treat Migraines
Pfizer to Acquire Seragon in $42.9B
PPVAE
Despite Highway Crisis, Many Worries Remain on US Oil Exports
Lululemon: Digital Sales Surge in 1Q Net Income, Revenue
Crisis of Market: US Stocks Climb; Nike Jumps

Table 8: Some generated conditional examples under condition *Business* (randomly sampled).

Failed Examples
Grammatical
Eat the service!
In addition, this location sucks it is.
Star Wars 7 will include US production on set
Conditional
(Negative) I was shocked that this is what I needed.
(Long) Are you actually drunk outside?
(Business) Michael Jackson's New Album 'Xscape'

Table 9: Some failed examples (cherry-picked).

6.5 Effect of Hyper-parameter β

Since β is an important hyper-parameter for PPVAE, we test $\beta \in \{0, 2, 5, 10\}$ on the long text generation task. From the results in Table 6, we find that β controls the balance between diversity and accuracy. Specifically, when β is too large, more diverse samples could be generated, but the accuracy may be sacrificed slightly. On the contrary, when β is too small, the accuracy could climb to a higher value, but meanwhile, the diversity drops drastically. Empirically, we find that $\beta = 5$ is an appropriate value for all tasks.

7 Case Study

We select some generated conditional text of each condition in Table 7. As shown in the table, our proposed PPVAE is capable of generating realistic conditional text. Also, shown in Table 8, on *topic* task, we randomly select some examples from the output of each model. The output of S-VAE seems to be diverse but is poorly conditioned. CTRL-GEN suffers an obvious diversity issue, which makes it repeatedly output similar text.

For the error analysis, we pick some failed examples of PPVAE in Table 9. We categorize the errors into two main classes. (1) **Grammatical**. Grammatical problems are common in NLG. As we analyze, this kind of errors can be mitigated with a deeper encoder and decoder with even more unlabeled data for pre-training. (2) **Conditional**. Conditional errors are of great interest to us since they lie in our focus. We choose three typical errors and list them in Table 9. In the first sentence, "shocked" is a subtle word which may indicate either positive or negative sentiment depending on the context. Thus, with a greedy decoding strategy, it may be incorrectly decoded into the other polarity. We believe this kind of errors could be fixed with more elaborate decoding strategies (e.g., Weighted Decoding (See et al., 2019)). In the second sentence, the length is limited by the nature of an interrogative sentence. As a linguistic fact, an interrogative sentence often has fewer words than a declarative sentence. In the third sentence, we remark an overlapping problem between classes. Some topics (e.g., music album) may appear in both business and entertainment news. In some way, these samples can also be considered as correctly conditioned ones, which highlights the importance of a fine-grained human evaluation on this task.

8 Conclusion

In this paper, we present a novel PPVAE framework for flexible conditional text generation, which decouples the text generation module from the condition representation module. The extensive experiments demonstrate the superiority of the proposed PPVAE against the existing alternatives on conditionality and diversity while allowing new conditions to be added without a full retraining.

Acknowledgments

We are grateful for the insightful comments from the anonymous reviewers. We would like to especially thank Daya Guo for his help and suggestions. This research was supported by National Natural Science Foundation of China (No. 61872278). Chenliang Li is the corresponding author.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *CoNLL*.

Andrew Brock, Jeff Donahue, and Karen Simonyan. 2019. Large scale GAN training for high fidelity natural image synthesis. In *ICLR*.

Tian Qi Chen, Xuechen Li, Roger B. Grosse, and David K. Duvenaud. 2018. Isolating sources of disentanglement in variational autoencoders. In *NeurIPS*.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *NeurIPS*.

Emilien Dupont. 2018. Learning disentangled joint continuous and discrete representations. In *NeurIPS*.

Jesse H. Engel, Matthew Hoffman, and Adam Roberts. 2018. Latent constraints: Learning to generate conditionally from unconditional generative models. In *ICLR*.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *ACL*.

Jessica Ficler and Yoav Goldberg. 2017. Controlling linguistic style aspects in neural language generation. *CoRR*, abs/1707.02633.

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. 2018. Style transfer in text: Exploration and evaluation. In *AAAI*.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *NeurIPS*.

Daya Guo, Yibo Sun, Duyu Tang, Nan Duan, Jian Yin, Hong Chi, James Cao, Peng Chen, and Ming Zhou. 2018. Question generation from SQL queries improves neural semantic parsing. In *EMNLP*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzkebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *ICML*.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Toward controlled generation of text. In *ICML*.

Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. 2019. CTRL: A conditional transformer language model for controllable generation. *CoRR*, abs/1909.05858.

Yuta Kikuchi, Graham Neubig, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. 2016. Controlling output length in neural encoder-decoders. In *EMNLP*.

Hyunjik Kim and Andriy Mnih. 2018. Disentangling by factorising. In *ICML*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Diederik P. Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. 2014. Semi-supervised learning with deep generative models. In *NeurIPS*.

Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *ICLR*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *NAACL-HLT*.

Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *ICML*.

Mehdi Mirza and Simon Osindero. 2014. Conditional generative adversarial nets. *CoRR*, abs/1411.1784.

Tong Niu and Mohit Bansal. 2018. Polite dialogue generation without parallel data. *Trans. Assoc. Comput. Linguistics*, 6:373–389.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Ali Razavi, Aäron van den Oord, and Oriol Vinyals. 2019. Generating diverse high-fidelity images with VQ-VAE-2. In *NeurIPS*.

Abigail See, Stephen Roller, Douwe Kiela, and Jason Weston. 2019. What makes a good conversation? how controllable attributes affect human judgments. In *NAACL-HLT*.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi S. Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *NeurIPS*.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. 2015. Learning structured output representation using deep conditional generative models. In *NeurIPS*.

Ilya O. Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schölkopf. 2018. Wasserstein auto-encoders. In *ICLR*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*.

Ke Wang and Xiaojun Wan. 2018. Sentigan: Generating sentimental texts via mixture adversarial networks. In *IJCAI*.

Liwei Wang, Alexander G. Schwing, and Svetlana Lazebnik. 2017. Diverse and accurate image description using a variational auto-encoder with an additive gaussian encoding space. In *NeurIPS*.

Jiqing Wu, Zhiwu Huang, Janine Thoma, Dinesh Acharya, and Luc Van Gool. 2018. Wasserstein divergence for gans. In *ECCV*.

Xiaopeng Yang, Xiaowen Lin, Shunda Suo, and Ming Li. 2018. Generating thematic chinese poetry using conditional variational autoencoders with hybrid decoders. In *IJCAI*.

Xiaoyuan Yi, Maosong Sun, Ruoyu Li, and Zonghan Yang. 2018. Chinese poetry generation with a working memory model. In *IJCAI*.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*.

Tiancheng Zhao, Ran Zhao, and Maxine Eskénazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *ACL*.

Wangchunshu Zhou, Tao Ge, Ke Xu, Furu Wei, and Ming Zhou. 2020. Self-adversarial learning with comparative discrimination for text generation. In *ICLR*.