Knowledge-Based Systems 256 (2022) 109766

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

PCAE: A framework of plug-in conditional auto-encoder for controllable text generation

Haoqin Tu [a], Zhongliang Yang [a,b,*], Jinshuai Yang [a], Siyu Zhang [a], Yongfeng Huang [a]

[a] Department of Electronic Engineering, Tsinghua University, Beijing, 100084, China
[b] Zhengzhou Xinda Institute of Advanced Technology, Zhengzhou, 450000, China

ARTICLE INFO

Article history:
Received 17 March 2022
Received in revised form 19 August 2022
Accepted 21 August 2022
Available online 6 September 2022

Keywords:
Controllable text generation
Plug-and-play
Model-agnostic
Transformers

ABSTRACT

Controllable text generation has taken a gigantic step forward these days. Yet existing methods are either constrained in a one-off pattern or not efficient enough for receiving multiple conditions at every generation stage. We propose a model-agnostic framework **P**lug-in **C**onditional **A**uto-**E**ncoder for Controllable Text Generation (**PCAE**) towards flexible and semi-supervised text generation. Our framework is "plug-and-play" with partial parameters to be fine-tuned in the pre-trained model (less than a half). Crucial to the success of PCAE is the proposed broadcasting label fusion network for navigating the global latent code to a specified local and confined space. Visualization of the local latent prior well confirms the primary devotion in hidden space of the proposed model. Moreover, extensive experiments across five related generation tasks (from 2 conditions up to 10 conditions) on both RNN-based and pre-trained BART [26] based auto-encoders reveal the high capability of PCAE, which enables generation that is highly manipulable, syntactically diverse and time-saving with minimum labeled samples. We will release our code at https://github.com/ImKeTT/pcae.

1. Introduction

Obtaining systems to automatically produce realistic-looking texts has been a goal pursued since the early stage of artificial intelligence [1]. In real life scenarios, to approach more human-like contexts, the generated sentences should be tailored to their specific audience [2]. As a result, controllable text generation (CTG) has drawn great attention nowadays [3–5]. Controllable text generation aims at generating coherent and grammatically correct texts whose attributes can be controlled [6], and/or abide by user-defined rules which reflect the particular interests of system users [2].

With the successful deployment of deep neural networks, recent proposed methods have brought us closer to this objective by producing texts with specified attributes. A general idea is to embed given conditions into an end-to-end training scheme [4,7,8] in order to produce sentences that fulfill given conditions, which has been illustrated in Fig. 1. Nevertheless, there are two main defects of these models that limit the application of these methods in reality. Firstly, these methods cannot deal well with real-world cases where conditions are not distributed at one time, i.e., new conditions for new using circumstance. In this scenario, models like SVAE [7], OPTIMUS [8] need to activate all model parameters to be trained for these new conditions, which are time-wasting, thus are not the ideal re-deployments for practical use [9]. Secondly, these models are mostly restricted to custom and well-designed language models, which means it is inconvenient to apply them directly to other more advanced language models for better modeling results. To address these problems for more practical application, another line for CTG follows the Pre-train and Plug-in (PnP) paradigm [5] has raised a lot research focus in recent years. By freezing the base language model (LM) and modifying few or no plug-in parameters, this paradigm is more flexible and powerful for controllable generation since it is parameter-efficient and can be applied to any advanced LM. Despite its success, there are two main defects regard to existing PnP works: one is that they are not convenient for creating texts with numerous categories at one time. Take currently the best-performed PnP language model PPVAE [10] as an example, when n conditions come in at some point, it demands to train additional n plug-in AEs to produce controlled texts. This drawback makes the whole system verbose and time-wasting when it meets a great amount of conditions. Another issue is that these PnP methods with only hidden mapping functions to be updated during plug-in process may be incapable of reaching a high degree of control. Since auto-encoders have shown favorable learning ability of text integral properties that are beneficial for controllable generation [11], we extend from existing text AE-based [12] PnP frameworks, and isolate the textual syntax module

* Corresponding author at: Department of Electronic Engineering, Tsinghua University, Beijing, 100084, China.
E-mail address: yangzl15@tsinghua.org.cn (Z. Yang).

https://doi.org/10.1016/j.knosys.2022.109766
0950-7051/

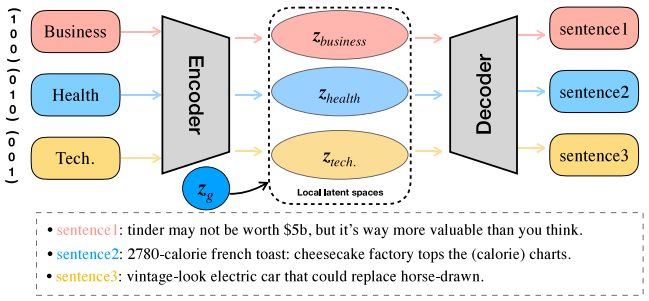


Fig. 1. A running example of the CTG task using auto-encoders. For controllable generation, we only need to input control signals (i.e., one-hot class label) and a global latent vector $\boldsymbol{z}_g$ sampled from standard Gaussian. Then the model produces texts that fulfill given conditions by creating specified local latent spaces.

from the input condition representation module by building the **BaseAE** and **PluginAE** separately.

Formally, the **BaseAE** can be any kind of text auto-encoder, which takes the main responsibility to formulize the basic sentence generation guidance as a standard LM. To benefit PluginAE in its high-dimensional hidden space, BaseAE is also expected to expatiate a robust and continuous latent manifold. As for **PluginAE**, it is a model-agnostic lightweight inserted component for BaseAE. Our PluginAE architecture is designed with efficient broadcasting label infuser *Broadcast Net*, incorporating label prior to BaseAE's latent space and enabling the plug-in model to learn all the conditions with one single training procedure. In purpose to achieve a higher level of control for our model, we choose to activate the decoder originated from the BaseAE during plug-in training. Our contributions can be listed as follow:

1. We explored a novel model-agnostic controllable text generation method PCAE. It is based on PnP framework and can be easily adopted to any kind of advanced auto-encoders for text controllable generation.
2. We devised the *Broadcast Net* for efficient fusion between conditions (labels) and latent space, so the model can generate controllable texts with very few labeled samples and time.
3. To explain the advantageous improvement of PCAE, we evaluated our model on five different related tasks with conditions ranging from 2 to 10. We further utilized both RNN-based and pre-trained BART [13] based auto-encoder to verify the effectiveness of proposed framework.

Inspiring results demonstrate that our model is both time-saving (reduce up to 35%) and highly controllable (near 90% accuracy with 100 labels for each class in the best case) compared with both competent RNN-based and BART-based baseline language models.

2. Related work

2.1. Text auto-encoders with latent variables

Latent variable models (LVM) have drawn massive attention in text generation field [10,12,14,15]. The latent space geometry of LVMs can conduct multiple view of knowledge in a given corpus (i.e., style, topic, and high-level linguistic or semantic features). There are two famous categories for text modeling with auto-encoders (AE), namely variational auto-encoders (VAE) [12] and adversarial auto-encoders (AAE) [16]. They commonly employ the evidence lower bound (ELBO) maximization of data $p_\theta(\boldsymbol{X})$ to update the holistic model. A major distinct between these two models lies in the regularization term of their ELBOs. While VAE takes a Kullback–Leibler (KL) penalty as its latent regulator, AAE introduces a discriminator to judge latent differences as illustrated below,

$$\log p(\boldsymbol{X}) \geq \underbrace{\mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{X})}[\log p(\boldsymbol{X} \mid \boldsymbol{z})]}_{\text{reconstruction term}} - \begin{cases} \underbrace{\mathbb{D}_{\mathrm{KL}}(q(\boldsymbol{z} \mid \boldsymbol{X}) \parallel p(\boldsymbol{z}))}_{\text{KL penalty}} & \text{ELBO of VAE} \\ \mathbb{E}_{p(\boldsymbol{z})}[-\log D(\boldsymbol{z})] & \text{ELBO of AAE} \\ \underbrace{+\mathbb{E}_{p(\boldsymbol{X})}[-\log(1 - D(E(\boldsymbol{X})))]}_{\text{Discriminator penalty}} & \end{cases}, \tag{1}$$

where function $D(\cdot)$ and $E(\cdot)$ for the ELBO of AAE denote its discriminator and encoder respectively. The VAE as a general tool is widely used in continuous generation (e.g., image generation). However, when it comes to the discrete domain (i.e., text generation), VAE is facing numerous plights, such as latent vacancy dilemma [17], latent vanishing problem [12], etc. The main reason is that VAE often neglects latent information provided by the encoder. In contrast to VAEs, AAEs maintain a strong coupling between their encoder and decoder, ensuring that the decoder does not ignore representations in the latent space, which makes it robust for latent knowledge interpretation and interpolation [16,18]. However, Li et al. [8] proved that a strong encoder such as pre-trained BERT in a VAE is very helpful to remit such issue. As a result, we employed AAE loss for RNN-based PCAE and VAE loss for pre-trained BART-based PCAE to show our framework is model-agnostic and effective under any auto-encoder.

2.2. Auto-encoders with pre-trained language models

Large pre-trained language models (PLMs) are gaining more and more popularity these days. With enormous resources being devoted, experienced encoders&decoders such as BERT [19], GPT-2 [20] and T5 [21] are devised to fully understand textual contents and create human-like sentences respectively. Incorporating these mighty PLMs as encoder and decoder of a variational auto-encoder can largely mitigate the KL collapse problem by offering the decoder a nonnegligible latent space from its encoder [8]. Several works to incorporate these PLMs into latent auto-encoders have been explored nowadays [8,22–25], which have shown promising potential in a varied multitude of tasks including unsupervised latent interpolation [8,24], controllable text generation [8] and prompt story generation [23], etc.

2.3. Controllable text generation

The core idea of controllable text generation is to generate textual contents with designated conditions to cope with specified circumstances and auditors. Formally, we follow the problem setting in previous works [4,10] to define the task: Given a set of k conditions $\boldsymbol{L} = \{l_1, l_2, \ldots, l_k\}$ (e.g., specific topics, sentiment labels), conditional text data $\boldsymbol{Y} = \{\boldsymbol{Y}_1, \boldsymbol{Y}_2, \ldots, \boldsymbol{Y}_k\}$ and unlabeled corpus $\boldsymbol{X}$, where each text corpus $\boldsymbol{Y}_i$ corresponds to its label l_i. With condition label l_i as input, we aim at learning a language model $\mathcal{F}(l_i)$ to calculate the distribution over the text samples $\boldsymbol{Y}_i$. Thus, when the condition l_i is specified, the model could generate realistic text samples that fulfill the given condition. And in practice, we usually leverage a trained text classifier to distinguish texts with different concepts (see Section 4.4.1 for controllability analysis).

To support generating sentences that fulfill such request, recent researches are mainly divided into threefold according to their training paradigm: supervised, self-supervised and semi-supervised. For fully supervised methods, adversarial components

Table 1
The main variable denotations in our method.

Variable	Description
$\boldsymbol{X}$	Input unlabeled text corpus
$\boldsymbol{Y}$	Input labeled text corpus
y_i	The ith word from a data point in $\boldsymbol{Y}$
$\boldsymbol{L}$	Task label set
l_i	The ith label from the label set
$\boldsymbol{Y_i}$	Labeled text corpus with label l_i
$\boldsymbol{Z_g}$	Global latent space
$\boldsymbol{z_g}$	Global latent vector from $\boldsymbol{Z_g}$
$\boldsymbol{Z_l}$	Local latent space
$\boldsymbol{z_l}$	Local latent vector from $\boldsymbol{Z_l}$
$\mathcal{F}_l$	Label embedding network
$\boldsymbol{e_{l_i}}$	Label embedding of label l_i
$\mathcal{F}_{z(t)}$	The tth latent transformation network
$\boldsymbol{z_{l(t)}}$	The local latent vector after $\mathcal{F}_{z(t)}$
$\boldsymbol{h_i}$	The ith hidden state of the decoder
$E(\cdot)$	The encoder of models
$D(\cdot)$	The latent discriminator of AAE models
$k(\cdot,\cdot)$	The kernel function
$p(\cdot)$	The prior distribution
$q(\cdot)$	The posterior distribution

like specified discriminators are widely employed [26,27]. In spite of their high controllability, they require abundant labeled data and enormous computational resources, which is unpractical for real world applications. Self-supervised methods commonly explore the hidden embeddings of LMs [15,27] and apply themselves to catch the underlying control rules during training, yet they normally provide sequences with a low degree of control.

The third party is semi-supervised, which requires only limited labeled data for controllable generation. SVAE [7] as the first semi-supervised VAE model, was initially applied to visual domain. Duan et al. [10] explored its modeling formulation into language domain, which treats the label embedding as an extended part of the latent variable when there are label-text pairs available. Li et al. [8] proposed OPTIMUS with BERT and GPT-2 as encoder and decoder respectively. They conducted controllable text generation via a latent space adversarial network using a two-stage training, which only requires labeled data at the second stage.

Apart from SVAE and OPTIMUS, one important branch named "Pre-train and Plug-in" (also known as plug-and-play) is rising recently. Since labeled samples are generally required only at "Plug-in" stage in PnP models, their training fashion is categorized as semi-supervised. Keskar et al. [28] used human-defined "control code" to pre-trained LMs in order to generate controllable texts, but needs full-scale fine-tuning. To reduce training time, [5] firstly proposed the concept of plug-and-play for conditional text generation, which generates controlled sentences by pulling the gradients of LMs along the desired path using extra components with few parameters. However, it was proposed based on large pre-trained language models and still requires hours to be trained. What followed was the PPVAE [10], which can be inserted to any pre-trained AE to create conditional texts. Nevertheless, it does not equip label infuser to incorporate condition knowledge explicitly into generation, thus has to train n plug-in VAEs when n new conditions come in. Naturally, when multiple conditions land at one time, PPVAE lacks the ability to deal with them elegantly and efficiently. Other lines of PnP controllable generation either targets at changing the prompts/prefix to be fed into the base LMs during training procedure [29,30], or shifting output probabilities from trained LMs at inference time [31,32]. These methods are mostly based on large pre-trained models and generally take hours to be fully tamed (sometimes their training times are even longer than fine-tuning) [30,31,33].

3. PCAE methodology

We present the main variable denotations in Table 1. The key idea of our framework is to reduce the resource consumption of training a language model with high controllability. The PnP framework with one full model training and plug-in controllable components is an efficient and flexible for this demand. Thus our model is separated into two disconnected sections: **BaseAE** and **PluginAE**, which corresponds to pre-training and plug-in training stage respectively. The model's workflow is in Fig. 2: the first figure represents the model structure of **BaseAE**, while the second figure is the structure of **PluginAE**. As for the third figure, it is the process for controllable text generation, which requires components from both **BaseAE** and **PluginAE**.

For pre-training stage, we use unlabeled textual data $\boldsymbol{X}$ to train the BaseAE language model (train from the scratch for RNN-based model and fine-tuning for BART-based model). For plug-in training, we input text-label pair $\{\boldsymbol{Y}, \boldsymbol{L}\} = \{\boldsymbol{Y}_i, l_i\}_i$, where $\boldsymbol{Y}_i$ is the training corpus from $\boldsymbol{Y}$ with label l_i. We use the labeled data pairs for conditional training in order to obtain the controllable decoder of PluginAE, which takes the latent variable and label condition l_i to generate controllable texts. Thus, once we trained the PluginAE, we only need to input the sampled global latent vector from its prior $\boldsymbol{z_g} \sim N(0, I)$ and a control label l_i (one-hot label) to the model for controlled generation. This training process makes PCAE only access to labels at the second stage, which makes it semi-supervised.

3.1. BaseAE pre-training/fine-tuning

BaseAE is under the obligation to present fluent and diverse texts and further derive meaningful latent representations. A BaseAE consists of three main components: the encoder, global latent space $\boldsymbol{Z_g}$ and the decoder. It should be noted that to ensure our BaseAE is a qualified LM, it ought to, in principle, be pre-trained on a very large text corpus (the unlabeled text data $\boldsymbol{X}$). As we employ both RNN and pre-trained BART in our framework, the pre-training stage of BaseAE represents training from the scratch for RNN and fine-tuning for BART-based model. For RNN-based BaseAE, we chose de-noising adversarial auto-encoder [34], whose loss function is the same shown in Eq. (1) except replacing $\boldsymbol{z}$ with global latent code $\boldsymbol{z_g} \sim \boldsymbol{Z_g}$. For BART-based BaseAE, we simply employed the BART encoder, decoder and the plain VAE training loss presented in Eq. (1) with $\boldsymbol{z}$ replaced with $\boldsymbol{z_g}$.

3.2. PluginAE plug-in training

PluginAE is the BaseAE model with additional plug-in components. It is the direct portal for controllable text generation, which forms a more structured local latent space with label signals and global latent space from BaseAE for controlled generation. To make the PluginAE controllable for generation, we send global latent embedding and label signal to neural networks to produce a locally structured latent space $\boldsymbol{Z_l}$, and further feed it to the decoder for generation.

Inspired by DenseNet [35], which densely connects neural representations with the latter layers of the network using skip connection [36], we utilize a label infuser that incorporates dense connection to sample the local variable $\boldsymbol{z_l}$ conditioned on both label and global latent information. That is to say, for given label representation, we broadcast it to every layer of a neural network using skip connection, which then produces the local latent vectors as output. We call this label infuser the *Broadcasting Net*. Specifically, for a *Broadcasting Net* with n layers, it takes the previous latent vectors (i.e., global latent vector $\boldsymbol{z_g}$ at the beginning) and any label l_i to generate the corresponding local latent vector $\boldsymbol{z_l}$, which can be formulized as:

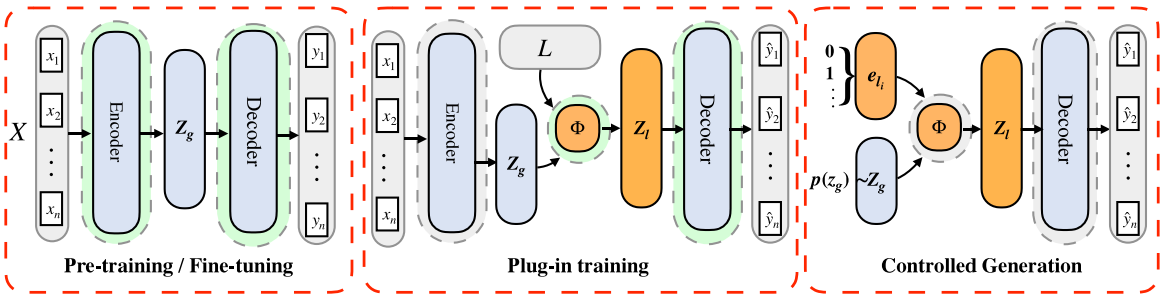


Fig. 2. A detailed workflow of the proposed framework. Parameters in components with green or gray backgrounds are activated or frozen respectively. During plug-in training, there is a label fusion function Φ (specifically the *Broadcasting Net*) for a more effective controlled sentence inference. During generation, we produce controllable texts by assigning a desired class label to the trained PluginAE.

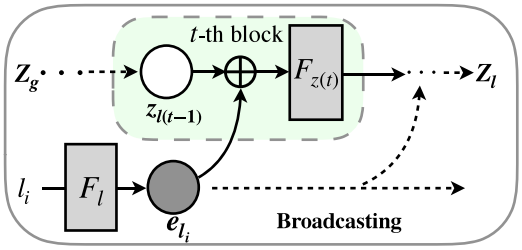


Fig. 3. The label fusion function Φ, which is essentially a broadcast operation between label embedding and latent codes. Thus it refers to the *Broadcasting Net*.

1. For input one-hot label l_i, we obtain its representation $\boldsymbol{e}_{\boldsymbol{l}_i}$ by label embedding layer $\mathcal{F}_l$.
2. Given $\boldsymbol{z}_{\boldsymbol{g}}$, we draw $\boldsymbol{z}_{\boldsymbol{l}}$ by sampling from the local latent distribution $p(\boldsymbol{z}_{\boldsymbol{l}} \mid \boldsymbol{z}_{\boldsymbol{g}}, \boldsymbol{e}_{\boldsymbol{l}_i})$. In detail, at tth layer, this transformation with $\boldsymbol{z}_{\boldsymbol{g}}$ and $\boldsymbol{e}_{\boldsymbol{l}_i}$ is implemented by a linear transformation $\mathcal{F}_{z(t)}$ after concatenation, so the overall modeling is formalized as:
$$\boldsymbol{z}_{\boldsymbol{l}} = \underbrace{\mathcal{F}_{z(n)}\left(\ldots\mathcal{F}_{z(1)}(\mathcal{F}_{z(0)}(\boldsymbol{z}_{\boldsymbol{g}} \oplus \boldsymbol{e}_{\boldsymbol{l}_i}) \oplus \boldsymbol{e}_{\boldsymbol{l}_i})\ldots \oplus \boldsymbol{e}_{\boldsymbol{l}_i}\right)}_{\text{totally } n \text{ Broadcasting Layers}},$$

Note that, to extend the broadcasting layer from one to multiple layers, we simply repeat it to broadcast the label signal to every layer of the label infuser as shown in Fig. 3. Hence we call this label infuser the *Broadcasting Net*.

3.3. Controllable text generation

Finally, we feed the sampled local latent vector $\boldsymbol{z}_{\boldsymbol{l}}$ to the decoder for word decoding. During training, the conditional modeling process of the decoder with labeled document $\boldsymbol{Y}$ can be formulized as:

$$\begin{aligned} p(\boldsymbol{Y} \mid \boldsymbol{z}_{\boldsymbol{l}}) &= p(y_1 \mid \boldsymbol{z}_{\boldsymbol{l}}) \prod_{i=2}^{n} p(y_i \mid y_{1:i-1}, \boldsymbol{z}_{\boldsymbol{l}}) \\ &= p(y_1 \mid \boldsymbol{z}_{\boldsymbol{l}}) \prod_{i=2}^{n} p(y_i \mid \boldsymbol{h}_i, \boldsymbol{z}_{\boldsymbol{l}}), \end{aligned} \tag{2}$$

where $\boldsymbol{h}_i$ is the ith hidden state of the decoder that satisfies $\boldsymbol{h}_i = \text{Decoder}(\boldsymbol{h}_{i-1}, y_{i-1}, \boldsymbol{z}_{\boldsymbol{l}})$. Unlike other PnP models that completely ignore **BaseAE** during plug-in training, we argue that to ensure the high efficiency of blending two separate domains (i.e., $\boldsymbol{e}_{\boldsymbol{l}}$ and $\boldsymbol{z}_{\boldsymbol{g}}$) and generating contexts with high quality from them, the decoder of BaseAE ought to take part in the optimization process and be regarded as a fine-tuning component in PluginAE. As a result, the reconstruction loss of PluginAE is on word token level instead of continuous latent level like PPVAE. This setting only activate less than a half parameters of BaseAE, and expends very little time (compared with baselines) but achieves considerable performances (see Section 4.4.3 for details).

Once we trained the PluginAE, we only need to input the sampled global latent vector from its prior $\boldsymbol{z}_{\boldsymbol{g}} \sim N(0, I)$ and a control label l_i to the model for controlled generation as shown in Fig. 2.

3.4. Training loss of PluginAE

For the plug-in training, we employ the labeled corpus $\boldsymbol{Y}$ as training data. Since the PluginAE inherits the training scheme of auto-encoders, its training loss consists of two parts, namely reconstruction loss and latent regularization term. To ensure the consistency of models' learning process, we employ adversarial auto-encoder (AAE) loss for RNN-based model and variational auto-encoder (VAE) loss for BART-based model respectively. The reconstruction loss of both types of model is the same, which is the cross-entropy loss between generated token logits and training sentences. The main difference between them is that, AAE loss uses the adversarial distance between latent prior and posterior for regularization, while VAE loss employs KL divergence. For RNN-based PluginAE, to avoid potential representation vanish issue in $\boldsymbol{z}_{\boldsymbol{l}}$ [37], we take a mutual information $I(\boldsymbol{Y}; \boldsymbol{z}_{\boldsymbol{l}})$ into consideration follow infoVAE [37,38]. It can be further factored in two items related to the KL divergence $\mathbb{D}_{\text{KL}}(q(\boldsymbol{z}_{\boldsymbol{l}} \mid \boldsymbol{Y}) \parallel p(\boldsymbol{z}_{\boldsymbol{l}}))$ and $\mathbb{D}_{\text{KL}}(q(\boldsymbol{z}_{\boldsymbol{l}}) \parallel p(\boldsymbol{z}_{\boldsymbol{l}}))$ as follow:

$$\begin{aligned} & I(\boldsymbol{Y}; \boldsymbol{z}_{\boldsymbol{l}}) \\ &= \int q(\boldsymbol{Y}, \boldsymbol{z}_{\boldsymbol{l}}) \log \frac{q(\boldsymbol{Y}, \boldsymbol{z}_{\boldsymbol{l}})}{q(\boldsymbol{Y}) q(\boldsymbol{z}_{\boldsymbol{l}})} d\boldsymbol{Y} d\boldsymbol{z}_{\boldsymbol{l}} \\ &= \int q(\boldsymbol{Y}, \boldsymbol{z}_{\boldsymbol{l}}) \log \frac{q(\boldsymbol{z}_{\boldsymbol{l}} \mid \boldsymbol{Y})}{q(\boldsymbol{z}_{\boldsymbol{l}})} d\boldsymbol{Y} d\boldsymbol{z}_{\boldsymbol{l}} \\ &= \int q(\boldsymbol{Y}, \boldsymbol{z}_{\boldsymbol{l}}) \left[\log \frac{q(\boldsymbol{z}_{\boldsymbol{l}} \mid \boldsymbol{Y})}{p(\boldsymbol{z}_{\boldsymbol{l}})} - \log \frac{q(\boldsymbol{z}_{\boldsymbol{l}})}{p(\boldsymbol{z}_{\boldsymbol{l}})}\right] d\boldsymbol{Y} d\boldsymbol{z}_{\boldsymbol{l}} \\ &= \int q(\boldsymbol{z}_{\boldsymbol{l}}, \boldsymbol{Y}) \log \frac{q(\boldsymbol{z}_{\boldsymbol{l}} \mid \boldsymbol{Y})}{p(\boldsymbol{z}_{\boldsymbol{l}})} d\boldsymbol{Y} d\boldsymbol{z}_{\boldsymbol{l}} - \int q(\boldsymbol{z}_{\boldsymbol{l}}) \log \frac{q(\boldsymbol{z}_{\boldsymbol{l}})}{p(\boldsymbol{z}_{\boldsymbol{l}})} d\boldsymbol{z}_{\boldsymbol{l}} \\ &= \mathbb{D}_{\text{KL}}\left[q(\boldsymbol{z}_{\boldsymbol{l}} \mid \boldsymbol{Y}) \parallel p(\boldsymbol{z}_{\boldsymbol{l}})\right] - \mathbb{D}_{\text{KL}}\left[q(\boldsymbol{z}_{\boldsymbol{l}}) \parallel p(\boldsymbol{z}_{\boldsymbol{l}})\right]. \end{aligned} \tag{3}$$

And we approximate the KL term $\mathbb{D}_{\text{KL}}(q(\boldsymbol{z}_{\boldsymbol{l}} \mid \boldsymbol{Y}) \parallel p(\boldsymbol{z}_{\boldsymbol{l}}))$ by adversarial distance for RNN-based PluginAE. Finally, the holistic objectives of both types of PluginAE are specified as follows:

$$\max p(\boldsymbol{Y}, l) \geq \mathbb{E}_{q(\boldsymbol{z}_{\boldsymbol{l}} \mid \boldsymbol{Y})}[\log p(\boldsymbol{Y} \mid \boldsymbol{z}_{\boldsymbol{l}})] - \lambda_{z_l} \mathcal{L}_{\boldsymbol{z}_{\boldsymbol{l}}}$$

where

$$\mathcal{L}_{\boldsymbol{z}_{\boldsymbol{l}}} = \begin{cases} Dist(\boldsymbol{z}_{\boldsymbol{l}}, E(\boldsymbol{Y})) \\ \quad - \lambda_{\text{info}} \mathbb{D}_{\text{KL}}(q(\boldsymbol{z}_{\boldsymbol{l}}) \parallel p(\boldsymbol{z}_{\boldsymbol{l}})) & \text{RNN-based} \\ \\ \mathbb{D}_{\text{KL}}(q(\boldsymbol{z}_{\boldsymbol{l}} \mid \boldsymbol{Y}) \parallel p(\boldsymbol{z}_{\boldsymbol{l}})) & \text{BART-based} \end{cases} \tag{4}$$

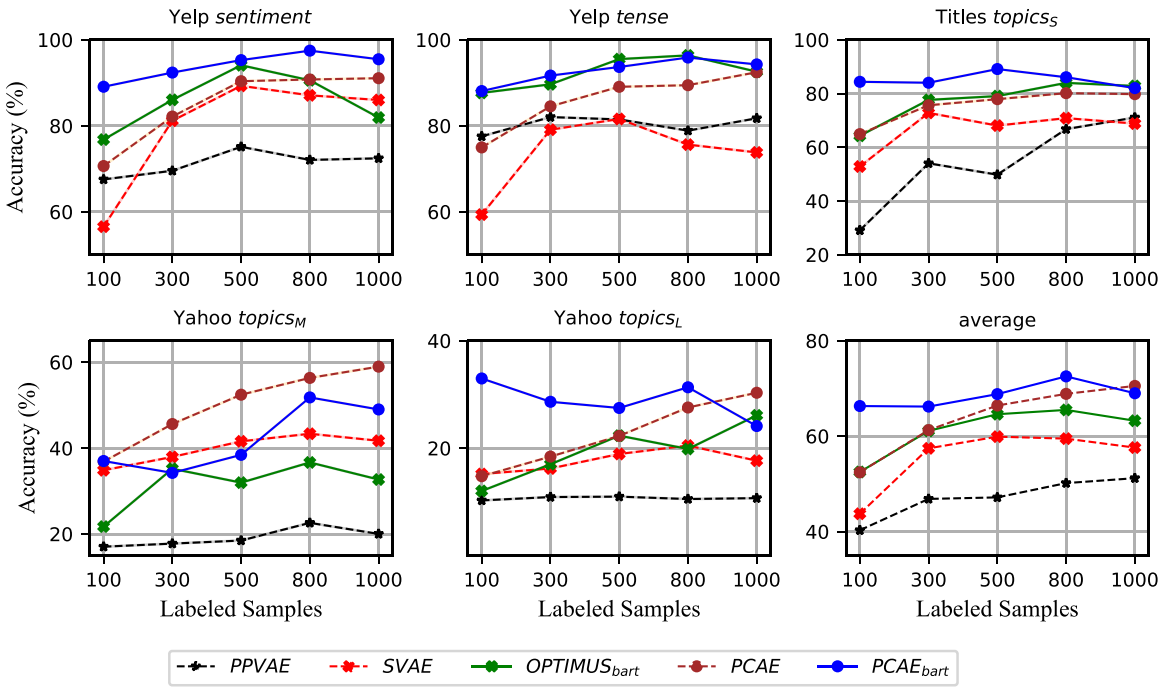


Fig. 4. Accuracy on five different tasks and averaged accuracy of them with varied number of labeled samples for each class. PCAE represents our framework under RNN framework with 12 layers in label fusion function Φ (*Broadcasting Net*). PCAE$_{\text{bart}}$ represents our framework under pre-trained BART with 10 layers in label fusion function. We implemented OPTIMUS [8] under the same pre-trained BART framework as ours (denote as OPTIMUS$_{\text{bart}}$).

Table 2
Statistics of the preprocessed datasets.

Dataset	#Voc. size	#Training Docs	#Validation Docs	#Test Docs	#Avg. length
Yelp	10,005	200,000	10,000	10,000	9.11
Titles	30,005	128,000	16,000	16,000	9.27
Yahoo	30,005	400,000	3000	3000	9.93

Here, the distance function in RNN-based loss is implemented as $Dist(\boldsymbol{z_l}, E(\boldsymbol{Y})) = \mathbb{E}_{p(\boldsymbol{Y},l)}[-\log(1 - D(E(\boldsymbol{Y})))] + \mathbb{E}_{p(\boldsymbol{z_g})}[-\log D(\boldsymbol{z_l})]$ with D, E to be the discriminator and encoder respectively. In practice, for RNN-based PCAE, we use multi-layer neural network (as the discriminator) to calculate its latent regularization term by classifying random noise and latent vectors. For BART-based PCAE, we use reparamerization trick [12] to parameterize the mean and log variance of the latent space (i.e., Gaussian) to calculate the KL divergence.

4. Experimental results and analysis

4.1. Implementation details

4.1.1. RNN-based implementation details

For BaseAE, we formally followed the settings in [34]. For three datasets to be trained, we mainly focused the short text generation and set the maximum vocabulary size to 10,000 for Yelp, 30,000 for Yahoo dataset and Titles dataset. Statistics for pre-training corpus are listed in Table 2. Word embedding dimension was 512. The encoder and decoder of BaseAE were bi-directional LSTM [39] and plain LSTM severally, and both with a hidden size of 1024 per direction. The size of global latent code was 128. The dimension of hidden state in the discriminator for latent distance measurement was set to 512. The noise function was taken from [34], we set word drop rate to 0.3. The weight of discriminator loss λ_{adv} was set to 10. For optimization, we utilized Adam [40] with learning rate 5×10^{-4} and a batch size of 256. We trained 50 epochs for three datasets, and stored model parameter weights on the epoch performs the best on validation set.

For PluginAE, the label embedding size n was 8. The label *Broadcasting Net* was implemented with pure linear functions and concatenations. During training, we activated the decoder, look-up linear function from decoder to word probability in the decoding section. And we also activated the linear function from latent codes to decoder embedding. As for optimizer, we selected Adam with learning rate 1×10^{-4} and a batch of 80 samples as input. We trained our model on each task until it converges. Through cross-validations, we chose the weight of adversarial loss λ_{adv} and latent regulator λ_{info} to be 30 and 50 respectively. For text generation, we chose categorical sampling with decoding temperature to be 0.8 according to ablation experiments.

As for the formal implementation of $\mathbb{E}_{p(\boldsymbol{Y},l)}[\mathbb{D}_{\text{KL}}(q(\boldsymbol{z_l} \mid \boldsymbol{Y}) \parallel p(\boldsymbol{z_l}))]$ approximation in the PluginAE training loss. We utilized another divergence Maximum Mean Discrepancy (MMD) [41,42] to efficiently optimize $\mathbb{D}_{\text{KL}}(q(\boldsymbol{z_l}) \parallel p(\boldsymbol{z_l}))$ term. MMD is widely used for quantifying the distance between two distributions using the kernel trick. The MMD between two distributions q and p is:

$$\begin{aligned}\mathbb{D}_{\text{MMD}}(q \parallel p) = &\mathbb{E}_{p(z),p(z')}\left[k\left(z, z'\right)\right] \\ &- 2\mathbb{E}_{q(z),p(z')}\left[k\left(z, z'\right)\right] \\ &+ \mathbb{E}_{q(z),q(z')}\left[k\left(z, z'\right)\right],\end{aligned} \tag{5}$$

the function $k(z, z')$ here is a definite kernel, and we chose it to be Gaussian.

4.1.2. BART-based implementation details

We utilized BART encoder and decoder as encoder and decoder of our AE model respectively. For BaseAE structure, we used the pre-trained tokenizer with the vocabulary size of 50,265. To perform BART under the paradigm of auto-encoder, a latent space is required to connect encoder and decoder during training. We derive this latent space using the mean pooling of the output of encoder and feed it to decoder as cross attention input, which is similar to OPTIMUS [8]. We loaded the medium-sized pre-trained

weight of BART[1] which consists of 6 transformer layers for encoder and decoder separately. The size of latent code was set to 128 like RNN-based ones. As for BaseAE training, we employed AdamW [43] optimizer with learning rate 1×10^{-4} for three corpus. During training, we followed OPTIMUS to utilized free KL threshold [44,45], the threshold was set to 0.1 and our model was trained with 4 cycles of KL annealing from 0 to 1 using cyclic KL annealing technique [46]. We finetuned the BART VAE with batch size 64 for 8, 10, 10 epochs for Yelp, Titles and Yahoo datasets respectively, which took around 2, 4 and 3 h to train.

For PluginAE structure, we only added label infuser based on BaseAE model. The structure of label infuser is exactly the same as RNN-based PCAE models. For PluginAE training, we trained the model with the VAE training objectives and set the free KL threshold to 0.1 for training consistency. During training, we activated the decoder and label Broadcasting components. As for optimizer, we selected AdamW with a batch of 32 samples as input. Moreover, since PLMs are sensitive to learning rate during training, we chose different learning rate for different tasks according to their classification performance, i.e., 1×10^{-4} for tense, topic_L tasks, 3×10^{-4} for sentiment, topic_S and topic_M tasks. We trained our model on each task until it achieves the highest controllability (i.e., the highest accuracy on each task). For text generation, we chose top-k nucleus sampling strategy [47] for decoding with $k = 50$ and $p = 1.0$ and sampling temperature to be 1.0. We generated 500 sentences for each class in every task for further evaluations.

4.1.3. Classifier implementation details

For the classifier we applied for text attribute classification, we employed a bi-directional LSTM with one layer and a hidden size of 256. The word embedding size was set to 128. As for optimization, we employed SGD [48] with learning rate 0.01. We trained the classifier on five tasks with 5000 labeled samples for each class and 50 for a batch until the loss converges. We saved the parameter weights of model performs the best on validation set during training.

4.2. Datasets

We conducted controlled experiments on different tasks to quantify the benefits of the various aspects of our approach. Specifically speaking, we followed previous works and carried out related tasks on three datasets: Yelp review [49], Titles [50] and Yahoo Question [51]. We chose five tasks from these three datasets, all with text semantic labels. Their detailed class descriptions are presented below:

- Yelp sentiment: 2 classes. Positive, Negative.
- Yelp tense: 2 classes. Present, Past. We make the same partition as described in [34].
- Titles topics_S: 4 classes. Business, Science & Technology, Entertainment, Health.
- Yahoo topics_M: 6 classes. Society & Culture, Science & Mathematics, Health, Education & Reference, Computers & Internet, Sports.
- Yahoo topics_L: 10 classes. 6 from Yahoo topics_M task with the addition to Business & Finance, Entertainment & Music, Family & Relationships, Politics & Government.

We respectively sampled 100, 300, 500, 800, 1000 examples from each classes for every task as plug-in training sets. The detailed statistical summary of three datasets for BaseAE pre-training is reported in Table 2.

[1] https://huggingface.co/facebook/bart-base.

4.3. Baselines

For RNN-based implementation of our model, the first comparison focuses on a model that, likes ours, is plug-and-play. PPVAE [10] generates controlled texts by feeding sentences with a heavy bias i.e., all from the same class. For multi-class controlled generation, PPVAE also takes negative samples (those not in the current desired category) and produce a negative loss to enhance its overall ability. That is to say, for n conditions (classes), PPVAE needs to produce n plug-in VAEs to be controlled, besides, at every training procedure it normally requires to encode all the given samples so that negative loss can be computed. Another baseline model SVAE [7] follows the end-to-end semi-supervised framework. It incorporates a classifier to provide conditional distribution for unlabeled data. Note that, to make it equal, we implement all baselines **under the same BaseAE**, and update the decoder parameters in PPVAE to make the competition impartial. Since other models like CTRL-GEN [4] has been proven much inferior in similar tasks [10], we did not take them for comparison.

For BART-based implementation of our model, we took OPTIMUS [8] as the baseline model. OPTIMUS is a strong baseline that reached unparallel language modeling ability including controllability, its controllable training requires a two-stage training process, and we used the same BaseAE of ours as its first stage tuning model out of fairness. OPTIMUS introduced a discriminator on latent space for adversarial training (to distinguish the real latent variable and Gaussian noise). We employed the same discriminator structure as in OPTIMUS with latent space dimension of 128. We denote BART-based OPTIMUS as OPTIMUS_{bart}.

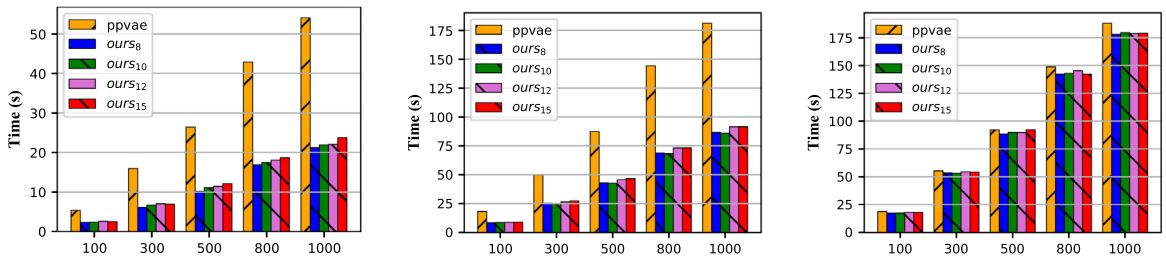
4.4. Evaluations and analysis

4.4.1. Controllability

To evaluate which degree of controlment the proposed model can obtain, we conducted experiments of text attribute classification. The classifier was trained with 5000 samples for each category. From the results in Fig. 4 and Table 3, we could draw the following conclusions: (1) Our RNN-based model is evidently superior to baseline PPVAE that follows the PnP paradigm in all circumstances. Also, in most situations, our model outperforms SVAE, which is an end-to-end model that needs a full-training every time. The last figure of averaged accuracy well confirms that PCAE is a better performer in controllable generation than baselines. (2) In some cases, the proposed model is capable of reaching comparatively high accuracy even with few available labels with either RNN or BART (e.g., RNN-based model reaches over 80% accuracy with only 300 labeled samples for each class on Yelp tense and sentiment). (3) When model's encoder and decoder are not very strong (i.e., RNN trained from the scratch), in tasks with too many categories (i.e., no less than 4 classes), models with explicit label signals (i.e., our model and SVAE) outperforms PPVAE distinctly and the proposed RNN-based model outperforms SVAE in most circumstances. We attribute these phenomenons to an effective label fusion network of ours compared with baselines. (4) BART-based models achieve generally higher accuracy than RNN-based models (except in topics_M task), which can be ascribed to applying powerful PLM encoder&decoder. (5) From the averaged result, among all baselines, the proposed PCAE_{bart} reaches the best performance in the most cases and comparable accuracy results in the rest situations.

4.4.2. Diversity

Intuitively, texts with high controllability often face with the conundrum of low diversity. AE-based works have been widely



(a) Yelp sentiment (b) Titles topics_{S} (c) Yahoo topics_{L}

Fig. 5. Training time (Y axis, counted in second) for plug-in modules with varied number of labeled samples (X axis) for each class on three tasks from three different datasets.

Table 3
Accuracy on five different tasks and averaged accuracy of them with varied number of labeled samples for each class. All model settings are the same as models in Fig. 4. We use **boldface** to indicate the best value.

Label Num.	Models	Sentiment	Tense	topics_{S}	topics_{M}	topics_{L}	Avg.
100	SVAE	56.53	59.34	52.86	34.85	15.18	43.75
	PPVAE	67.53	77.56	29.10	17.06	10.26	40.30
	OPTIMUS_{bart}	76.80	87.67	64.29	21.73	12.01	52.44
	PCAE	70.65	74.98	64.92	36.86	14.77	52.44
	PCAE_{bart}	**89.10**	**88.10**	**84.40**	**37.00**	**32.97**	**66.31**
300	SVAE	81.20	79.10	72.82	37.95	16.21	57.46
	PPVAE	68.55	82.04	54.05	17.78	10.89	46.86
	OPTIMUS_{bart}	86.05	89.67	77.67	35.20	17.00	61.28
	PCAE	82.16	84.54	75.67	**45.61**	18.42	61.28
	PCAE_{bart}	**92.40**	**91.70**	**84.05**	34.23	**28.63**	**66.20**
500	SVAE	89.31	81.60	68.10	41.58	18.91	59.90
	PPVAE	75.12	81.49	49.82	18.49	10.97	47.18
	OPTIMUS_{bart}	94.10	**95.53**	79.08	31.97	22.32	66.42
	PCAE	90.39	89.09	77.94	**52.43**	22.28	66.42
	PCAE_{bart}	**95.30**	93.70	**89.15**	38.43	**27.47**	**68.81**
800	SVAE	87.10	75.63	70.88	43.34	20.48	59.48
	PPVAE	72.07	78.89	66.78	22.61	10.52	50.17
	OPTIMUS_{bart}	90.60	**96.40**	84.00	36.67	19.84	68.87
	PCAE	90.79	89.46	80.17	**56.34**	27.56	68.87
	PCAE_{bart}	**97.50**	95.90	**86.10**	51.80	**31.33**	**72.53**
1000	SVAE	86.02	73.82	68.85	41.71	17.66	57.61
	PPVAE	72.44	81.76	71.10	20.03	10.69	51.21
	OPTIMUS_{bart}	81.93	92.60	**82.85**	32.69	26.10	63.23
	PCAE	91.09	92.48	79.80	**58.96**	**30.31**	**70.52**
	PCAE_{bart}	**95.50**	**94.30**	82.05	49.00	24.11	68.99

explored and revealed the capacity to generate diverse contents [52,53]. We measured distinct n-grams (normalized by the length of text) as in [54]:

$$\text{Distinct-}n = \frac{\text{unique } n\text{-grams}}{N}, \tag{6}$$

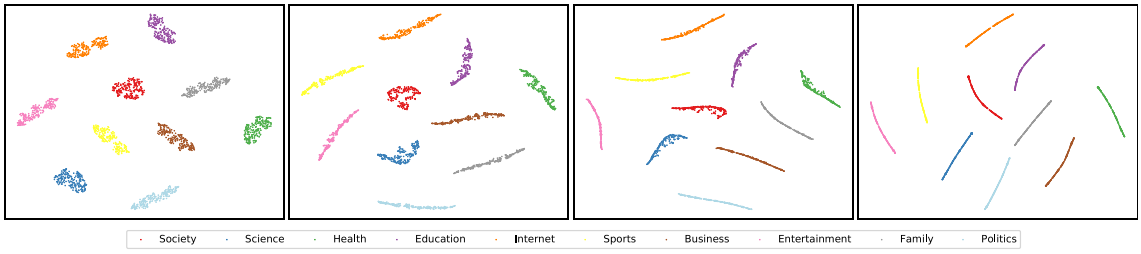
where N is the number of generated words. Higher the Distinctive scores are, less likely the model produces "dull texts". We report the ratio of unique 1-gram and 2-gram values (refer to D-1 and D-2 respectively) of 5k sentences for each category from any model on five tasks in Table 4. (1) For both RNN-based and BART-based models, our model is able to generate more diverse sentences than baselines in most cases. Especially on topics_{M} and topics_{L}, the gains in diversity from our model to other methods are significantly higher (PCAE is twice as better as PPVAE or SVAE on D-1, PCAE_{bart} is almost triple as better as OPTIMUS_{bart} on D-2). (2) Introducing pre-trained BART to our model can largely bring up the D-1 values. We ascribe it to the larger vocabulary size of pre-trained BART (i.e., 50,265 for BART by default and maximum 30,000 for RNNs by presetting). (3) BART-based PCAE has little or non improvement on D-2 compared with RNN-based models. This is because PCAE_{bart} reaches a higher degree of controllability, which is naturally contradicts to text diversity. That is, as the increase of *broadcasting layer*, PCAE_{bart} may produce more structured local latent space, which is in favor of generating controllable sentences but against high diversity in the latent representations. This can be explained as the $\text{PCAE}_{bart\,10}$ always generates more diverse texts than $\text{PCAE}_{bart\,15}$ in the table.

4.4.3. Training cost

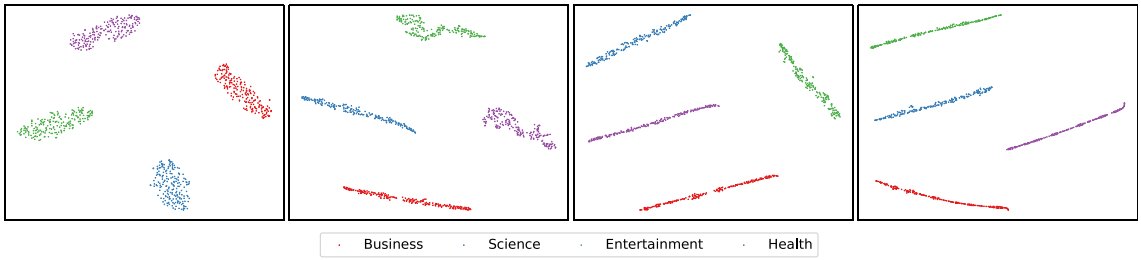
Can our model be applicable in real scenarios? To answer that question, the training time of plug-in modules should be paid great attention to. We make comparison between RNN-based PCAE and PPVAE, which are plug-and-play. In detail, we recorded times that every model consumed until it converged, and all models were trained on the same machine with one TITAN X GPU successively. For every picture from Fig. 5, we draw the time consumption in seconds of RNN-based models. We have the following conclusions: (1) Our model outperforms PPVAE distinctively in all circumstances (less than a half of the training time compared with PPVAE in both model settings). This demonstrates the effectiveness of our model for being a more pragmatic tool for controllable text generation. (2) In cases of two tasks from Yahoo dataset, PPVAE is less disadvantageous in time cost than other tasks compared with ours. We argue that these cases should be analyzed combined with the accuracy metric:

Table 4
Distinct-1 and Distinct-2 (refer to D-1 and D-2 respectively) of different models with varied number of labeled samples for each class. Real values are the presented ones divided by 100. **PCAE**$_n$ and **PCAE**$_{bartn}$ means the proposed RNN-based PCAE and BART-based PCAE model with n broadcasting layers respectively. We use **boldface** to indicate the best value.

Label Num.	Models	Sentiment		Tense		topics_S		topics_M		topics_L	
		D-1	D-2	D-1	D-2	D-1	D-2	D-1	D-2	D-1	D-2
100	SVAE	1.38	20.68	2.01	26.54	1.13	23.33	0.29	**9.29**	0.12	4.87
	PPVAE	2.91	27.60	3.93	**35.99**	1.59	17.49	0.19	4.07	0.18	3.43
	OPTIMUS_{bart}	7.39	17.62	6.11	12.27	4.28	10.90	0.12	0.31	0.45	2.73
	PCAE_{10}	3.14	31.19	3.52	33.98	2.49	23.46	0.27	6.22	0.40	4.99
	PCAE_{15}	2.92	**31.38**	3.47	33.63	2.81	**25.11**	0.28	6.22	0.39	4.96
	PCAE_{bart10}	9.98	25.54	**11.06**	28.34	**5.43**	11.53	**0.30**	1.46	**1.28**	**7.45**
	PCAE_{bart15}	**10.41**	27.17	9.63	25.69	4.42	7.94	0.17	1.05	1.13	5.29
300	SVAE	1.23	22.86	1.64	31.36	0.93	20.47	0.20	**7.02**	0.11	4.82
	PPVAE	2.50	28.75	3.75	35.42	1.34	16.71	0.19	3.91	0.17	3.40
	OPTIMUS_{bart}	9.39	24.07	11.10	24.15	**5.65**	14.06	0.13	0.49	0.60	3.22
	PCAE_{10}	2.31	28.18	2.95	35.23	1.96	23.70	0.21	6.50	0.33	5.26
	PCAE_{15}	2.58	28.93	2.99	**37.06**	1.93	**23.89**	0.21	6.47	0.33	5.20
	PCAE_{bart10}	**14.08**	**40.31**	**12.74**	22.53	5.59	16.73	**0.36**	1.82	1.47	7.85
	PCAE_{bart15}	12.86	35.94	12.08	19.76	4.80	11.07	0.17	0.90	**4.14**	**9.98**
500	SVAE	1.34	25.23	1.83	34.55	0.78	19.19	0.16	5.89	0.11	4.73
	PPVAE	3.53	34.83	3.73	35.23	1.13	15.90	0.18	3.76	0.17	3.37
	OPTIMUS_{bart}	10.96	26.37	11.84	24.74	6.52	15.58	0.08	0.29	0.89	4.11
	PCAE_{10}	2.19	28.65	2.69	31.60	2.15	24.52	0.23	6.43	0.38	5.03
	PCAE_{15}	2.30	34.86	2.66	**36.32**	2.10	**25.28**	0.23	**6.59**	0.37	5.05
	PCAE_{bart10}	14.21	40.91	**17.72**	33.30	8.65	19.53	0.38	1.87	**1.60**	**7.67**
	PCAE_{bart15}	**15.41**	**42.58**	7.16	11.14	**8.98**	17.51	**0.39**	1.92	0.64	3.55
800	SVAE	1.75	25.42	1.95	35.32	0.71	19.35	0.15	5.32	0.10	4.69
	PPVAE	2.32	31.71	3.57	34.83	0.99	15.26	0.18	3.62	0.17	3.34
	OPTIMUS_{bart}	12.39	31.06	14.34	32.25	7.13	17.43	0.10	0.31	0.43	2.64
	PCAE_{10}	2.12	29.71	2.96	32.62	2.52	24.27	0.26	6.39	0.44	5.45
	PCAE_{15}	2.25	34.05	2.80	**37.74**	2.45	**24.90**	0.27	**6.45**	0.44	5.48
	PCAE_{bart10}	**15.64**	46.79	**15.51**	28.60	**9.95**	22.25	**0.40**	2.14	**1.66**	**8.14**
	PCAE_{bart15}	15.38	**48.19**	15.34	29.41	6.99	14.90	0.12	0.75	0.81	4.31



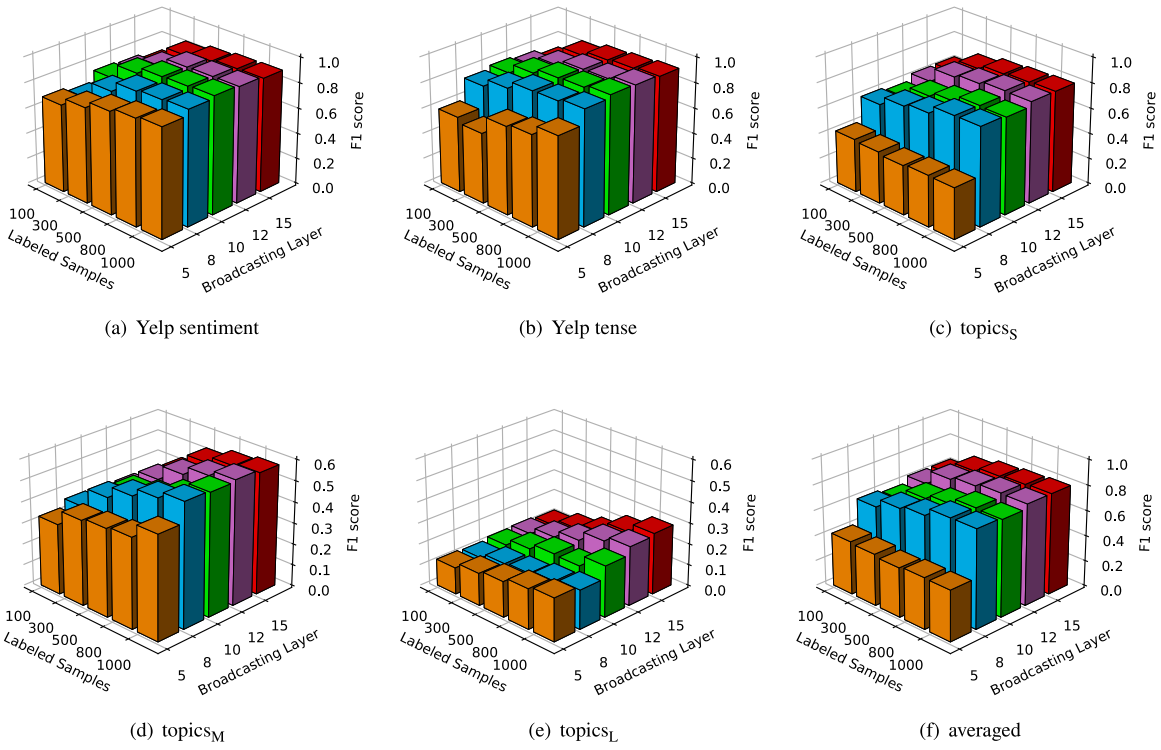
(a)



(b)

Fig. 6. Visualization of the designed prior of local latent $\boldsymbol{z_l}$ from RNN-based PCAE with varied layers in *Broadcasting Net* on (a) Yahoo topics_L task and (b) topics_S task. The number of broadcasting layer is chosen from 5, 8, 10, 12 in succession.

PPVAE cannot handle situations where too many classes come in at one time, thus quits learning to be controlled (e.g., PPVAE holds steady accuracy of 20% and 15% for topics_M and topics_L in Fig. 4) and converges without acquiring enough knowledge from the biased data. (3) Varying the number of broadcasting layers in the label infuser does not influence the training time apparently. Because each broadcasting layer contains a small number of parameters to be updated. We also present the averaged time



(a) Yelp sentiment (b) Yelp tense (c) topics_{S}

(d) topics_{M} (e) topics_{L} (f) averaged

Fig. 7. Macro-F1 score for generated controllable text classification with regard to varying broadcasting layer and labeled samples for each class based on RNN-based PCAE.

Table 5
Averaged time cost for training with 300 labeled data for each class on RNN-based systems.

Models	Time cost	
SVAE	∼1.1 h (every time)	
PPVAE	1 h (only once)	55.71 s (Plug-in)
Ours	1 h (only once)	**37.18 s** (Plug-in)

cost of baseline models compared with our model in Table 5. SVAE and OPTIMUS_{bart} as end-to-end model and two-stage fine-tuning model respectively are much slower and unpractical to be controllable when coming across too many conditions.

4.4.4. Justification for latent optimization

To illustrate that the whole plug-in training process is more inclined to enhance the controlled expression in the latent field (i.e., helps $\boldsymbol{z_g}$ towards meaningful and structured $\boldsymbol{z_l}$), updating decoder is only an auxiliary tool for producing more fluent sequential content. We resort to visualize the input of model decoder using T-SNE [55], the visualized input corresponds to the prior of local latent code $\boldsymbol{z_l}$. As shown in Fig. 6, under the setting of RNN-based PCAE on topics_{L} task and topics_{S} task, the factitious local latent prior with given labels are well separated. Priors from *Broadcasting Net* with added layers show more compact and structured clustering, which further verifies our claim in Section 4.4.2 for the discussions about generated text diversity. This phenomenon also justifies that the *Broadcasting Net* as our label infuser function Φ efficiently elevates the capacity in the learnt hidden representations. From another view, PCAE produces controllable sentences by focusing on $\boldsymbol{z_g}$ to $\boldsymbol{z_l}$ rather than updating the parameters in decoder. We can also observe that, topics that are intuitively more correlated in real-life are actually closer in clustering (e.g., society is always in the center, family locates close to education

4.4.5. Generated sentences

We present texts with different conditions on three tasks (sentiment, tense, topics_{S}) in Table 6. For RNN-based PCAE, presented sentences intuitively match the given attribute well. For instance, in Yelp tense task, context assigned with "Past" attribute has signal words of "was" and "expected". In Titles topics_{S} task, sentence that belong to different conditions owns keywords including the names of celebrities ("kardashia", "beyoncé"), name of cell phones ("iphone") or code for diseases ("ebola virus"). For BART-based PCAE, sentences are more complex and fluent. For instance, in Yelp sentiment task, sentence with "Positive" label presents "good", "smooth" and "tasty" in the comment. As in Titles topics_{S} task, sentences given different topics show their own features, such as text belongs to the "Business" topic talks about the company value of Tinder, while text from "Health" focuses on the calorie of food.

4.4.6. Ablation study

We conducted all ablation experiments on RNN-based PCAE and similar behavior should be observed for BART-based one. Firstly, we specifically analysis the impact of different broadcasting layer and labeled samples for training to the control of generated texts on five tasks. As shown in Fig. 7, (1) More broadcasting layers can bring up the control ability of the proposed model until it is 12, that is 10 or 12 layers of *Broadcasting Net* generally perform the best among all presented experiments results. (2) With the number of labels increases, the F1 scores climb higher in general. And keep increasing the number of labeled samples will finally make the our model in full supervision.

Secondly, we explore the effectiveness of latent regulator $\mathcal{L}_{\text{info}}$. Theoretically speaking, the latent regulator i.e., mutual information maximization term between code $\boldsymbol{z_l}$ and observed data $\boldsymbol{X}$ can bring the model to a deeper learning stage, thus helpful in enhancing the controllable ability of our model. To verify the impact of such latent regulator, we use test F1 score on sentiment and tense tasks with 100 and 800 labeled sample for each

Table 6
Conditional examples generated by PCAE on three tasks for qualitative analysis.

	Task	Condition	Generated examples
RNN-based	Sentiment	Negative	• this main experience was totally short and no good.
		Positive	• good experience in the area
	Tense	Past	• the fries were interesting, but not overcooked by the toppings.
		Present	• the place is always good and the brunch is the beginning.
	topics_S	Entertainment	• kate gomez sparks justin bieber engagement in an ring
		Business	• linkedin to oracle will offer hiding on data today
		Technology	• lg launches mini spots of a now $199 on the September
		Health	• sierra of un: scientists fight ebola virus family
BART-based	Sentiment	Negative	• furthermore, the food quality does not meet the price.
		Positive	• really good burgers and the oak stout beer is really smooth and tasty.
	Tense	Past	• the paella was really really really good and the manhattans did not disappoint.
		Present	• they compete with each other and **do** not care about you!
	topics_S	Entertainment	• is hollywood to blame? or journalism? a battle on twitter
		Business	• tinder may not be worth $5b, but its way more valuable than you think
		Technology	• vintage-look electric car that could replace horse-drawn carriages
		Health	• 2780-calorie french toast: cheesecake factory tops the (calorie) charts

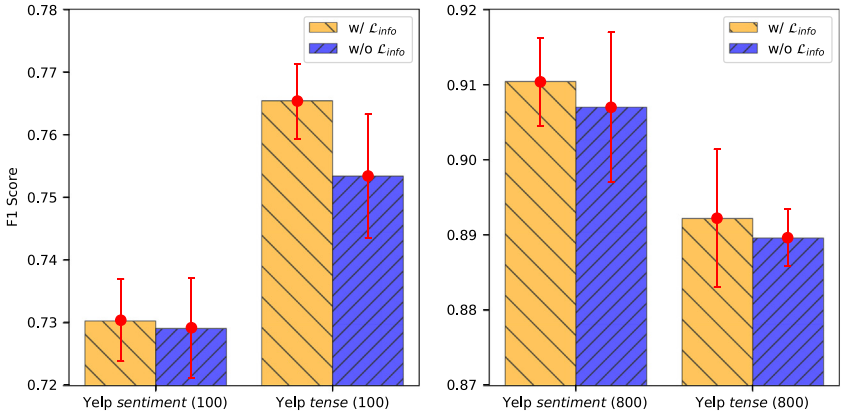


Fig. 8. Ablation study with F1 indicator of RNN-based PCAE with 12 layers in terms of latent regulator $\mathcal{L}_{\text{info}}$ on sentiment and tense tasks. Number in parentheses behind task name is the label number for each class for model training.

categories respectively, further train PluginAE with or without $\mathcal{L}_{\text{info}}$ and report the F1 score in Fig. 8. From the results, we can see that (1) the latent regulator helps strengthen control capability of our model. (2) F1 results of models with $\mathcal{L}_{\text{info}}$ gain a narrower standard deviation than models without $\mathcal{L}_{\text{info}}$, which demonstrates that the latent regulator enhances the robustness of PCAE.

5. Conclusion

In this work, we present a model-agnostic semi-supervised controllable text generation framework PCAE, which leads to remarkable empirical performance on both RNN-based and pre-trained BART-based settings. By adding *Broadcasting Net* to existing plug-and-play system, we put this mainstay framework into a more universal pattern to generate controllable textual contents with auto-encoders. The visualization of learned local latent prior justifies the model optimization takes place in the hidden space. More importantly, with higher degree of controllability and competitive diversity of output texts plus less resource costs to apply the proposed framework, PCAE shows promising results on validating the effectiveness of proposed *Broadcasting Net*. To make our model more practical in real-world scenarios, we apply both plain RNN trained from the scratch and pre-trained BART to our framework, and consistent results proves the effectiveness of the proposed framework.

CRediT authorship contribution statement

Haoqin Tu: Methodology, Software, Formal analysis, Writing – original draft. **Zhongliang Yang:** Conceptualization, Validation, Data curation, Writing – review & editing. **Jinshuai Yang:** Validation, Visualization, Writing – review & editing. **Siyu Zhang:** Validation, Investigation, Writing – review & editing. **Yongfeng Huang:** Supervision, Resources, Investigation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported in part by the National Key Research and Development Program of China (No. 2021ZD0113902), the National Natural Science Foundation of China (No. U1936208 and No. 61862002, No. U1936216), China Postdoctoral Science Foundation (No. 2021M701943, No. 2022T150370), Open Foundation of Henan Key Laboratory of Cyberspace Situation Awareness, China (No. HNTS2022008).

References

[1] J.R. Meehan, TALE-SPIN, an interactive program that writes stories, in: Ijcai, Vol. 77, 1977, p. 9198.

[2] C. Garbacea, Q. Mei, Neural language generation: Formulation, methods, and evaluation, 2020, arXiv preprint arXiv:2007.15780.
[3] J. Ficler, Y. Goldberg, Controlling linguistic style aspects in neural language generation, EMNLP 2017 (2017) 94.
[4] Z. Hu, Z. Yang, X. Liang, R. Salakhutdinov, E.P. Xing, Toward controlled generation of text, in: International Conference on Machine Learning, PMLR, 2017, pp. 1587–1596.
[5] S. Dathathri, A. Madotto, J. Lan, J. Hung, E. Frank, P. Molino, J. Yosinski, R. Liu, Plug and play language models: A simple approach to controlled text generation, 2019, arXiv preprint arXiv:1912.02164.
[6] M. Elhadad, Constraint-based text generation using local constraints and argumentation to generate a turn in conversation, 1990.
[7] D.P. Kingma, S. Mohamed, D.J. Rezende, M. Welling, Semi-supervised learning with deep generative models, in: Advances in Neural Information Processing Systems, 2014, pp. 3581–3589.
[8] C. Li, X. Gao, Y. Li, B. Peng, X. Li, Y. Zhang, J. Gao, Optimus: Organizing sentences via pre-trained modeling of a latent space, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2020, pp. 4678–4699.
[9] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, S. Gelly, Parameter-efficient transfer learning for NLP, in: International Conference on Machine Learning, PMLR, 2019, pp. 2790–2799.
[10] Y. Duan, C. Xu, J. Pei, J. Han, C. Li, Pre-train and Plug-in: Flexible conditional text generation with variational auto-encoders, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2019, pp. 253–262.
[11] L. Fang, C. Li, J. Gao, W. Dong, C. Chen, Implicit deep latent variable models for text generation, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 2019, pp. 3946–3956.
[12] S. Bowman, L. Vilnis, O. Vinyals, A. Dai, R. Jozefowicz, S. Bengio, Generating sentences from a continuous space, in: Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, 2016, pp. 10–21.
[13] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, L. Zettlemoyer, BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020, pp. 7871–7880.
[14] W. Wang, Z. Gan, H. Xu, R. Zhang, G. Wang, D. Shen, C. Chen, L. Carin, Topic-guided variational auto-encoder for text generation, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), 2019, pp. 166–177.
[15] H. Tang, M. Li, B. Jin, A topic augmented text generation model: Joint learning of semantics and structural features, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 2019, pp. 5090–5099.
[16] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, B. Frey, Adversarial autoencoders, 2015, arXiv preprint arXiv:1511.05644.
[17] P. Xu, J.C.K. Cheung, Y. Cao, On variational learning of controllable representations for text without supervision, in: International Conference on Machine Learning, PMLR, 2020, pp. 10534–10543.
[18] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, L. Bottou, Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion, J. Mach. Learn. Res. 11 (12) (2010).
[19] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), 2019, pp. 4171–4186.
[20] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al., Language models are unsupervised multitask learners, OpenAI Blog 1 (8) (2019) 9.
[21] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P.J. Liu, et al., Exploring the limits of transfer learning with a unified text-to-text transformer., J. Mach. Learn. Res. 21 (140) (2020) 1–67.
[22] D. Liu, G. Liu, A transformer-based variational autoencoder for sentence generation, in: 2019 International Joint Conference on Neural Networks, IJCNN, IEEE, 2019, pp. 1–7.
[23] L. Fang, T. Zeng, C. Liu, L. Bo, W. Dong, C. Chen, Transformer-based conditional variational autoencoder for controllable story generation, 2021, arXiv preprint arXiv:2101.00828.
[24] S. Park, J. Lee, Finetuning pretrained transformers into variational autoencoders, in: Proceedings of the Second Workshop on Insights from Negative Results in NLP, 2021, pp. 29–35.
[25] H. Tu, Z. Yang, J. Yang, S. Zhang, Y. Huang, Adavae: Exploring adaptive GPT-2s in variational auto-encoders for language modeling, 2022, arXiv preprint arXiv:2205.05862.
[26] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, P. Abbeel, Infogan: Interpretable representation learning by information maximizing generative adversarial nets, in: Proceedings of the 30th International Conference on Neural Information Processing Systems, 2016, pp. 2180–2188.
[27] K. Wang, X. Wan, SentiGAN: Generating sentimental texts via mixture adversarial networks, in: IJCAI, 2018, pp. 4446–4452.
[28] N.S. Keskar, B. McCann, L.R. Varshney, C. Xiong, R. Socher, Ctrl: A conditional transformer language model for controllable generation, 2019, arXiv preprint arXiv:1909.05858.
[29] E. Wallace, S. Feng, N. Kandpal, M. Gardner, S. Singh, Universal adversarial triggers for attacking and analyzing NLP, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 2019, pp. 2153–2162.
[30] X.L. Li, P. Liang, Prefix-Tuning: Optimizing continuous prompts for generation, in: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), 2021, pp. 4582–4597.
[31] B. Krause, A.D. Gotmare, B. McCann, N.S. Keskar, S. Joty, R. Socher, N.F. Rajani, GeDi: Generative discriminator guided sequence generation, in: Findings of the Association for Computational Linguistics: EMNLP 2021, 2021, pp. 4929–4952.
[32] D. Pascual, B. Egressy, F. Bolli, R. Wattenhofer, Directed beam search: Plug-and-play lexically constrained language generation, 2020, arXiv preprint arXiv:2012.15416.
[33] J. He, C. Zhou, X. Ma, T. Berg-Kirkpatrick, G. Neubig, Towards a unified view of parameter-efficient transfer learning, in: International Conference on Learning Representations, 2021.
[34] T. Shen, J. Mueller, R. Barzilay, T. Jaakkola, Educating text autoencoders: Latent representation guidance via denoising, in: International Conference on Machine Learning, PMLR, 2020, pp. 8719–8729.
[35] G. Huang, Z. Liu, L. Van Der Maaten, K.Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 4700–4708.
[36] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.
[37] S. Zhao, J. Song, S. Ermon, Infovae: Information maximizing variational autoencoders, 2017, arXiv preprint arXiv:1706.02262.
[38] A.L. Rezaabad, S. Vishwanath, Learning representations by maximizing mutual information in variational autoencoders, in: 2020 IEEE International Symposium on Information Theory, ISIT, IEEE, 2020, pp. 2729–2734.
[39] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[40] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.
[41] A. Gretton, K. Borgwardt, M. Rasch, B. Schölkopf, A. Smola, A kernel method for the two-sample-problem, Adv. Neural Inf. Process. Syst. 19 (2006) 513–520.
[42] Y. Li, K. Swersky, R. Zemel, Generative moment matching networks, in: International Conference on Machine Learning, PMLR, 2015, pp. 1718–1727.
[43] I. Loshchilov, F. Hutter, Decoupled weight decay regularization, in: International Conference on Learning Representations, 2018.
[44] B. Li, J. He, G. Neubig, T. Berg-Kirkpatrick, Y. Yang, A surprisingly effective fix for deep latent variable modeling of text, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 2020, pp. 3603–3614.
[45] T. Pelsmaeker, W. Aziz, Effective estimation of deep generative language models, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020, pp. 7220–7236.
[46] H. Fu, C. Li, X. Liu, J. Gao, A. Celikyilmaz, L. Carin, Cyclical annealing schedule: A simple approach to mitigating KL vanishing, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), 2019, pp. 240–250.
[47] A. Holtzman, J. Buys, L. Du, M. Forbes, Y. Choi, The curious case of neural text degeneration, in: International Conference on Learning Representations, 2019.
[48] L. Bottou, Large-scale machine learning with stochastic gradient descent, in: Proceedings of COMPSTAT'2010, Springer, 2010, pp. 177–186.
[49] T. Shen, T. Lei, R. Barzilay, T. Jaakkola, Style transfer from non-parallel text by cross-alignment, 2017, arXiv preprint arXiv:1705.09655.

[50] Z. Fu, X. Tan, N. Peng, D. Zhao, R. Yan, Style transfer in text: Exploration and evaluation, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 32, 2018.
[51] Z. Yang, Z. Hu, R. Salakhutdinov, T. Berg-Kirkpatrick, Improved variational autoencoders for text modeling using dilated convolutions, in: International Conference on Machine Learning, PMLR, 2017, pp. 3881–3890.
[52] L. Wang, A.G. Schwing, S. Lazebnik, Diverse and accurate image description using a variational auto-encoder with an additive Gaussian encoding space, in: NIPS, 2017.
[53] A. Razavi, A. van den Oord, O. Vinyals, Generating diverse high-fidelity images with vq-vae-2, in: Advances in Neural Information Processing Systems, 2019, pp. 14866–14876.
[54] J. Li, M. Galley, C. Brockett, J. Gao, W.B. Dolan, A diversity-promoting objective function for neural conversation models, in: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2016, pp. 110–119.
[55] L. Van der Maaten, G. Hinton, Visualizing data using t-SNE, J. Mach. Learn. Res. 9 (11) (2008).