Generating Sentences from Disentangled Syntactic and Semantic Spaces

Yu Bao[1*] **Hao Zhou**[2*] **Shujian Huang**[1†] **Lei Li**[2] **Lili Mou**[3]
Olga Vechtomova[3] **Xinyu Dai**[1] **Jiajun Chen**[1]
[1]National Key Laboratory for Novel Software Technology, Nanjing University, China
[2]ByteDance AI Lab, Beijing, China
[3]University of Waterloo, Canada
{baoy,huangsj,dxy,chenjj}@nlp.nju.edu.cn
{zhouhao.nlp,lileilab}@bytedance.com
doublepower.mou@gmail.com, ovechtomova@uwaterloo.ca

Abstract

Variational auto-encoders (VAEs) are widely used in natural language generation due to the regularization of the latent space. However, generating sentences from the continuous latent space does not explicitly model the syntactic information. In this paper, we propose to generate sentences from disentangled syntactic and semantic spaces. Our proposed method explicitly models syntactic information in the VAE's latent space by using the linearized tree sequence, leading to better performance of language generation. Additionally, the advantage of sampling in the disentangled syntactic and semantic latent spaces enables us to perform novel applications, such as the unsupervised paraphrase generation and syntax-transfer generation. Experimental results show that our proposed model achieves similar or better performance in various tasks, compared with state-of-the-art related work.

1 Introduction

Variational auto-encoders (VAEs, Kingma and Welling, 2014) are widely used in language generation tasks (Serban et al., 2017; Kusner et al., 2017; Semeniuta et al., 2017; Li et al., 2018b). VAE encodes a sentence into a probabilistic latent space, from which it learns to decode the same sentence. In addition to traditional reconstruction loss of an autoencoder, VAE employs an extra regularization term, penalizing the Kullback–Leibler (KL) divergence between the encoded posterior distribution and its prior. This property enables us to sample and generate sentences from the continuous latent space. Additionally, we can even manually manipulate the latent space, inspiring various applications such as sentence interpolation (Bowman et al., 2016) and text style transfer (Hu et al., 2017).

However, the continuous latent space of VAE blends syntactic and semantic information together, without modeling the syntax explicitly. We argue that it may be not necessarily the best in the text generation scenario. Recently, researchers have shown that explicitly syntactic modeling improves the generation quality in sequence-to-sequence models (Eriguchi et al., 2016; Zhou et al., 2017; Li et al., 2017; Chen et al., 2017). It is straightforward to adopt such idea in the VAE setting, since a vanilla VAE does not explicitly model the syntax. A line of studies (Kusner et al., 2017; Gómez-Bombarelli et al., 2018; Dai et al., 2018) propose to impose context-free grammars (CFGs) as hard constraints in the VAE decoder, so that they could generate syntactically valid outputs of programs, molecules, etc.

However, the above approaches cannot be applied to syntactic modeling in VAE's continuous latent space, and thus, we do not enjoy the two benefits of VAE, namely, *sampling* and *manipulation*, towards the syntax of a sentence.

In this paper, we propose to generate sentences from a disentangled syntactic and semantic spaces of VAE (called DSS-VAE). DSS-VAE explicitly models syntax in the continuous latent space of VAE, while retaining the sampling and manipulation benefits. In particular, we introduce two continuous latent variables to capture semantics and syntax, respectively. To separate the semantic and syntactic information from each other, we borrow the adversarial approaches from the text style-transfer research (Hu et al., 2017; Fu et al., 2018; John et al., 2018), but adapt it into our scenario of syntactic modeling. We also observe that syntax and semantics are highly interwoven, and therefore further propose an adversarial reconstruction loss to regularize the syntactic and se-

[*]Equal contributions.
[†]Corresponding author.

mantic spaces.

Our proposed DSS-VAE takes following advantages:

First, explicitly syntactic modeling in VAE's latent space improves the quality of unconditional language generation. Experiments show that, compared with traditional VAE, DSS-VAE generates more fluent sentences (lower perplexity), while preserving more amount of encoded information (higher BLEU scores for reconstruction). Comparisons with a state-of-the-art syntactic language model (Shen et al., 2017) are also included.

Second, the advantage of manipulation in the syntactic and semantic spaces of DSS-VAE provides a natural way of unsupervised paraphrase generation. If we sample a vector in the syntactic space but perform max *a posterior* (MAP) inference in the semantic space, we are able to generate a sentence with the same meaning but different syntax. This is known as *unsupervised paraphrase generation*, as no parallel corpus is needed during training. Experiments show that DSS-VAE outperforms the traditional VAE as well as a state-of-the-art Metropolis-Hastings sampling approach (Miao et al., 2019) in this task.

Additionally, with the disentangled syntactic and semantic latent spaces, we propose an interesting application that transfers the syntax of one sentence to another. Both qualitative and quantitative experimental results show that DSS-VAE could graft the designed syntax to another sentence under certain circumstances.

2 Related Work

The variational auto-encoders (VAEs) is proposed by Kingma and Welling (2014) for image generation. Bowman et al. (2016) successfully applied VAE in the NLP domain, showing that VAE improves recurrent neural network (RNN)-based language modeling (RNN-LM, Mikolov et al., 2010); that VAE allows sentence sampling and sentence interpolation in the continuous latent space. Later, VAE is widely used in various natural language generation tasks (Gupta et al., 2018; Kusner et al., 2017; Hu et al., 2017; Deriu and Cieliebak, 2018).

Syntactic language modeling, to the best of our knowledge, could be dated back to Chelba (1997). Charniak (2001) and Clark (2001) propose to utilize a top-down parsing mechanism for language modeling. Dyer et al. (2016) and Kuncoro et al. (2017) introduce the neural network to this direction. The Parsing-Reading-Predict Network (PRPN, Shen et al., 2017), which reports a state-of-the-art results on syntactic language modeling, learns a latent syntax by training with a language modeling objective. Different from their work, our approach models syntax in a continuous space, facilitating sampling and manipulation of syntax.

Our work is also related to style-transfer text generation (Fu et al., 2018; Li et al., 2018a; John et al., 2018). In previous work, the style is usually defined by categorical features such as sentiment. We move one step forward, extending their approach to the sequence level and dealing with more complicated, non-categorical syntactic spaces. Due to the complication of syntax, we further design adversarial reconstruction losses to encourage the separation of syntax and semantics.

3 Approach

In this section, we present our proposed DSS-VAE in detail. We first introduce the variational autoencoder in §3.1. Then, we describe the general architecture of DSS-VAE in §3.2, where we explain how we generate sentences from disentangled syntactic and semantic latent spaces and how we disentangle information from the two separated spaces. Model training is discussed in §3.3.

3.1 Variational Autoencoder

A traditional VAE employs a probabilistic latent variable $\boldsymbol{z}$ to encode the information of a sentence $\boldsymbol{x}$, and then decodes the original $\boldsymbol{x}$ from $\boldsymbol{z}$. The probability of a sentence $\boldsymbol{x}$ could be computed as:

$$p(\boldsymbol{x}) = \int p(\boldsymbol{z}) p(\boldsymbol{x}|\boldsymbol{z})\,\mathrm{d}\boldsymbol{z} \tag{1}$$

where $p(\boldsymbol{z})$ is the prior, and $p(\boldsymbol{x}|\boldsymbol{z})$ is given by the decoder. VAE is trained by maximizing the *evidence lower bound* (ELBO):

$$\begin{aligned}&\log p(\boldsymbol{x}) \geq \text{ELBO} \\ &= \mathop{\mathbb{E}}_{q(\boldsymbol{z}|\boldsymbol{x})} \left[\log p(\boldsymbol{x}|\boldsymbol{z})\right] - \text{KL}\left(q(\boldsymbol{z}|\boldsymbol{x}) \,\|\, p(\boldsymbol{z})\right)\end{aligned} \tag{2}$$

3.2 Proposed Method: DSS-VAE

Our DSS-VAE is built upon the vanilla VAE, but extends Eqn. (1) by adopting two separate latent variables $\boldsymbol{z}_{\text{sem}}$ and $\boldsymbol{z}_{\text{syn}}$ to capture semantic and syntactic information, respectively. Specifically, we assume that the probability of a sentence $\boldsymbol{x}$ in DSS-VAE could be computed as:

$$p(\boldsymbol{x}) = \int p(\boldsymbol{z}_{\text{sem}}, \boldsymbol{z}_{\text{syn}}) p(\boldsymbol{x}|\boldsymbol{z}_{\text{sem}}, \boldsymbol{z}_{\text{syn}})\, \mathrm{d}\boldsymbol{z}_{\text{sem}}\, \mathrm{d}\boldsymbol{z}_{\text{syn}}$$
$$= \int p(\boldsymbol{z}_{\text{sem}}) p(\boldsymbol{z}_{\text{syn}}) p(\boldsymbol{x}|\boldsymbol{z}_{\text{sem}}, \boldsymbol{z}_{\text{syn}})\, \mathrm{d}\boldsymbol{z}_{\text{sem}}\, \mathrm{d}\boldsymbol{z}_{\text{syn}}$$

where $p(\boldsymbol{z}_{\text{sem}})$ and $p(\boldsymbol{z}_{\text{syn}})$ are the priors; both are set to be independent multivariate Gaussian $\mathcal{N}(\mathbf{0}, \mathrm{I})$.

Similar to (2), we optimize the *evidence lower bound* (ELBO) for training:

$$\begin{aligned}\log p(\boldsymbol{x}) \geq\ & \text{ELBO} \\ =\ & \mathop{\mathbb{E}}_{q(\boldsymbol{z}_{\text{sem}}|\boldsymbol{x}) q(\boldsymbol{z}_{\text{syn}}|\boldsymbol{x})} \left[\log p(\boldsymbol{x}|\boldsymbol{z}_{\text{sem}}, \boldsymbol{z}_{\text{syn}})\right] \\ & - \text{KL}\left(q(\boldsymbol{z}_{\text{sem}}|\boldsymbol{x}) \,\|\, p(\boldsymbol{z}_{\text{sem}})\right) \\ & - \text{KL}\left(q(\boldsymbol{z}_{\text{syn}}|\boldsymbol{x}) \,\|\, p(\boldsymbol{z}_{\text{syn}})\right)\end{aligned}$$

where $q(\boldsymbol{z}_{\text{sem}}|\boldsymbol{x})$ and $q(\boldsymbol{z}_{\text{syn}}|\boldsymbol{x})$ are posteriors for the two latent variables. We further assume the variational posterior families, $q(\boldsymbol{z}_{\text{sem}}|\boldsymbol{x})$ and $q(\boldsymbol{z}_{\text{syn}}|\boldsymbol{x})$, are independent, taking the form $\mathcal{N}(\boldsymbol{\mu}_{\text{sem}}, \boldsymbol{\sigma}^2_{\text{sem}})$ and $\mathcal{N}(\boldsymbol{\mu}_{\text{syn}}, \boldsymbol{\sigma}^2_{\text{syn}})$, respectively, We use RNN to parameterize the posteriors (also called the *encoder*). Here, $\boldsymbol{\mu}_{\text{sem}}$, $\boldsymbol{\sigma}_{\text{sem}}$, $\boldsymbol{\mu}_{\text{syn}}$, and $\boldsymbol{\sigma}_{\text{syn}}$ are predicted by the encoder network, described as follows.

Encoding In the encoding phase, we first obtain the sentence representation $\boldsymbol{r}_x$ by an RNN with the gated recurrent units (GRUs, Cho et al., 2014); then, $\boldsymbol{r}_x$ is evenly split into two spaces $\boldsymbol{r}_x = [\boldsymbol{r}_x^{\text{sem}}; \boldsymbol{r}_x^{\text{syn}}]$.

For the semantic encoder, we compute the mean and variance of $q(\boldsymbol{z}_{\text{sem}}|\boldsymbol{x})$ from $\boldsymbol{r}_x^{\text{sem}}$ as:

$$\begin{bmatrix} \boldsymbol{\mu}_{\text{sem}} \\ \boldsymbol{\sigma}_{\text{sem}} \end{bmatrix} = \begin{bmatrix} W^{\mu}_{\text{sem}} \\ W^{\sigma}_{\text{sem}} \end{bmatrix} \text{ReLU}(W_{\text{sem}} \boldsymbol{r}_x^{\text{sem}} + \boldsymbol{b}_{\text{sem}})$$

where the activation function is the rectified linear unit (ReLU, Nair and Hinton, 2010). W^{μ}_{sem},W^{σ}_{sem},W_{sem}, and $\boldsymbol{b}_{\text{sem}}$ are the parameters of the semantic encoder.

Likewise, a syntactic encoder predicts $\boldsymbol{\mu}_{\text{syn}}$ and $\boldsymbol{\sigma}_{\text{syn}}$ for $q(\boldsymbol{z}_{\text{syn}}|\boldsymbol{x})$ in the same way, with parameters W^{μ}_{syn},W^{σ}_{syn},W_{syn}, and $\boldsymbol{b}_{\text{syn}}$.

Decoding in the Training Phase We first sample from the posterior distributions by the reparameterization trick (Kingma and Welling, 2014), obtaining sampled semantic and syntactic representations, $\boldsymbol{z}_{\text{sem}}$ and $\boldsymbol{z}_{\text{syn}}$; then, they are concatenated as $\boldsymbol{z} = [\boldsymbol{z}_{\text{sem}}; \boldsymbol{z}_{\text{syn}}]$ and fed as the initial state of the decoder for reconstruction.

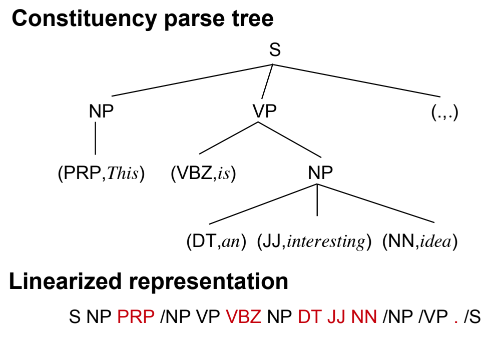

Figure 1: The parse tree and its linearized tree sequence of a sentence "*This is an interesting idea.*"

Decoding in the Test Phase The treatment depends on applications. If we would like to synthesize a sentence from scratch, both $\boldsymbol{z}_{\text{syn}}$ and $\boldsymbol{z}_{\text{sem}}$ are sampled from prior. If we would like to preserve/vary semantics/syntax, max a posterior (MAP) inference or sampling could be applied in respective spaces. Details are provided in § 4.

In the following part, we will introduce how syntax is modeled in our approach and how syntax and semantics are ensured to be separated.

3.2.1 Modeling Syntax by Predicting Linearized Tree Sequence

While previous studies have tackled the problem of categorical sentiment modeling in the latent space (Hu et al., 2017; Fu et al., 2018), syntax is much more complicated and not finitely categorical. We propose to adopt the linearized tree sequence to explicitly model syntax in the latent space of VAE.

Figure 1 shows the constituency parse tree of the sentence "*This is an interesting idea.*" The linearized tree sequence can be obtained by traversing the syntactic tree in a top-down order; if the node is non-terminal, we add a backtracking node (e.g., `/NP`) after its child nodes are traversed.

We ensure that $\boldsymbol{z}_{\text{syn}}$ contains syntactic information by predicting the linearized tree sequence. In training, the parse tree for sentences are obtained by the `ZPar`[1] toolkit, and serves as the groundtruth training signals; in testing, we do not need external syntactic trees. We build an RNN (independent of the VAE's decoder) to predict such linearized parse trees, where each parsing token is represented by an embedding (similar to a traditional RNN decoder). Notice that, a node and

[1]https://www.sutd.edu.sg/cmsresource/faculty/yuezhang/zpar.html

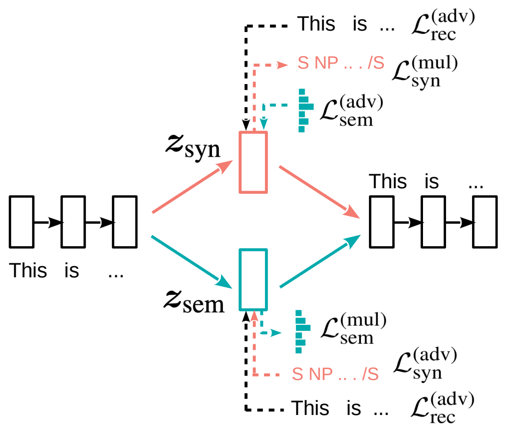
This is ... $\mathcal{L}_{\text{rec}}^{(\text{adv})}$
S NP .. . /S $\mathcal{L}_{\text{syn}}^{(\text{mul})}$
$\mathcal{L}_{\text{sem}}^{(\text{adv})}$
z_{syn}
This is ...
This is ...
z_{sem}
$\mathcal{L}_{\text{sem}}^{(\text{mul})}$
S NP .. . /S $\mathcal{L}_{\text{syn}}^{(\text{adv})}$
This is ... $\mathcal{L}_{\text{rec}}^{(\text{adv})}$

Figure 2: Overview of our DSS-VAE. Forward dashed arrows are multi-task losses; backward dashed arrows are adversarial losses.

its backtracking, e.g., `NP` and `/NP`, have different embeddings.

The linearized tree sequence has achieved promising parsing results in a traditional constituency parsing task (Vinyals et al., 2015; Liu et al., 2018; Vaswani et al., 2017), which shows its ability of preserving syntactic information. Additionally, the linearized tree sequence works in a sequence-to-sequence fashion, so that it can be used to regularize the latent spaces.

3.2.2 Disentangling Syntax and Semantics into Different Latent Spaces

Having solved the problem of syntactic modeling, we now turn to the question: how could we disentangle syntax and semantics from each other?

We are inspired by the research in text style transfer and apply auxiliary losses to regularize the latent space (Hu et al., 2017; Fu et al., 2018).

In particular, we adopt the multi-task and adversarial losses in John et al. (2018), but extend it to the sequence level. In §3.2.3, we further propose two adversarial reconstruction losses to discourage the model to encode a sentence from a single subspace.

Multi-Task Loss Intuitively, a multi-task loss ensures that each space (z_{syn} or z_{sem}) should capture respective information.

For the semantic space, we predict the bag-of-words (BoW) distribution of a sentence from z_{sem} with softmax, whose objective is the cross-entropy loss against the groundtruth distribution $\boldsymbol{t}$, given by:

$$\mathcal{L}_{\text{sem}}^{(\text{mul})} = -\sum\nolimits_{w\in\mathcal{V}} t_w \log p(w|z_{\text{sem}}) \qquad (3)$$

where $p(w|z_{\text{syn}})$ is the predicted distribution. BoW has been explored by previous work (Weng et al., 2017; John et al., 2018), showing good ability of preserving semantics.

For the syntactic space, the multi-task loss trains a model to predict syntax on z_{syn}. Due to our proposal in §3.2.1, we could build a dedicated RNN, predicting the tokens in the linearized parse tree sequence, whose loss is:

$$\mathcal{L}_{\text{syn}}^{(\text{mul})} = -\sum\nolimits_{i=1}^{n} \log p(s_i|s_1\cdots s_{i-1}, z_{\text{syn}}) \quad (4)$$

where s_i is a token in the linearized parse tree (with a total length of n).

Adversarial Loss The adversarial loss is widely used for aligning samples from different distributions. It has various applications, including style transfer (Hu et al., 2017; Fu et al., 2018; John et al., 2018) and domain adaptation (Tzeng et al., 2017). To apply adversarial losses, we add extra model components (known as *adversaries*) to predict semantic information t_w based on the syntactic space z_{syn}, but to predict syntactic information $s_1\cdots s_{n-1}$ based on the semantic space z_{sem}. They are denoted by $p_{\text{adv}}(w|z_{\text{syn}})$ and $p_{\text{adv}}(s_i|s_1\cdots s_{i-1}, z_{\text{sem}})$.

The training of these adversaries are similar to (3) and (4), except that the gradient only trains the adversaries themselves, and does not back-propagate to VAE.

Then, VAE is trained to "fool" the adversaries by maximizing their losses, i.e., minimizing the following terms:

$$\mathcal{L}_{\text{sem}}^{(\text{adv})} = \sum\nolimits_{w\in\mathcal{V}} t_w \log p_{\text{adv}}(w|z_{\text{syn}}) \qquad (5)$$

$$\mathcal{L}_{\text{syn}}^{(\text{adv})} = \sum\nolimits_{i=1}^{n} \log p_{\text{adv}}(s_i|s_1\cdots s_{i-1}, z_{\text{sem}}) \quad (6)$$

In this phase, the adversaries are fixed and their parameters are not updated.

3.2.3 Adversarial Reconstruction Loss

Our next intuition is that syntax and semantics are more interwoven to each other than other information such as style and content.

Suppose, for example, the syntax and semantics have been perfectly separated by the losses in §3.2.2, where z_{sem} could predict BoW well, but does not contain any information about the syntactic tree. Even in this ideal case, the decoder can reconstruct the original sentence from z_{sem} by simply learning to re-order words (as z_{sem} does contain BoW). Such word re-ordering knowledge is indeed learnable (Ma et al., 2018), and does not necessarily contain the syntactic information. Therefore, the multi-task and adversarial losses for syntax and semantics do not suffice to regularize DSS-VAE.

We now propose an *adversarial reconstruction loss* to discourage the sentence being predicted by a single subspace z_{syn} or z_{sem}. When combined, however, they should provide a holistic view of the entire sentence. Formally, let $\boldsymbol{z}_s$ be a latent variable ($\boldsymbol{z}_s = z_{\text{syn}}$ or z_{sem}). A decoding adversary is trained to predict the sentence based on $\boldsymbol{z}_s$, denoted by $p_{\text{rec}}(x_i|x_1 \cdots x_{i-1}, \boldsymbol{z}_s)$. Then, the adversarial reconstruction loss is imposed by minimizing

$$\mathcal{L}_{\text{rec}}^{(\text{adv})}(\boldsymbol{z}_s) = \sum\nolimits_{i=1}^{M} \log p_{\text{rec}}(x_i|x_{<i}, \boldsymbol{z}_s) \quad (7)$$

Such adversarial reconstruction loss is applied to both the syntactic and semantic spaces, shown by black bashed arrows in Figure 2.

3.3 Training Details

Overall Training Objective The overall training loss is a combination of the VAE loss (2), the multi-task and adversarial losses for syntax and semantics (3–6), as well as the adversarial reconstruction losses (7), , i.e., minimizing

$$\begin{aligned}
\mathcal{L} &= \mathcal{L}_{\text{vae}} + \mathcal{L}_{\text{aux}} \\
&= - \mathop{\mathbb{E}}_{q(\boldsymbol{z}_{\text{sem}}|\boldsymbol{x})q(\boldsymbol{z}_{\text{syn}}|\boldsymbol{x})} \log \left[p(\boldsymbol{x}|\boldsymbol{z}_{\text{sem}}, \boldsymbol{z}_{\text{syn}})\right] \\
&\quad + \lambda_{\text{sem}}^{\text{KL}} \, \text{KL}\left(q(\boldsymbol{z}_{\text{sem}}|\boldsymbol{x}) \,\|\, p(\boldsymbol{z}_{\text{sem}})\right) \\
&\quad + \lambda_{\text{syn}}^{\text{KL}} \, \text{KL}\left(q(\boldsymbol{z}_{\text{syn}}|\boldsymbol{x}) \,\|\, p(\boldsymbol{z}_{\text{syn}})\right) \\
&\quad + \lambda_{\text{sem}}^{\text{mul}} \mathcal{L}_{\text{sem}}^{(\text{mul})} + \lambda_{\text{sem}}^{\text{adv}} \mathcal{L}_{\text{sem}}^{(\text{adv})} + \lambda_{\text{sem}}^{\text{rec}} \mathcal{L}_{\text{rec}}^{(\text{adv})}(\boldsymbol{z}_{\text{sem}}) \\
&\quad + \lambda_{\text{syn}}^{\text{mul}} \mathcal{L}_{\text{syn}}^{(\text{mul})} + \lambda_{\text{syn}}^{\text{adv}} \mathcal{L}_{\text{syn}}^{(\text{adv})} + \lambda_{\text{syn}}^{\text{rec}} \mathcal{L}_{\text{rec}}^{(\text{adv})}(\boldsymbol{z}_{\text{syn}})
\end{aligned} \quad (8)$$

where the $\lambda_{\text{sem}}^{\text{KL}}$, $\lambda_{\text{syn}}^{\text{KL}}$, $\lambda_{\text{sem}}^{\text{mul}}$, $\lambda_{\text{sem}}^{\text{adv}}$, $\lambda_{\text{sem}}^{\text{rec}}$, $\lambda_{\text{syn}}^{\text{mul}}$, $\lambda_{\text{syn}}^{\text{adv}}$, and $\lambda_{\text{syn}}^{\text{rec}}$ are the hyperparameters to adjust the importance of each loss in overall objective.

Hyperparameter Tuning We select the parameter values with the lowest ELBO value on the validation set in all experiments. They are tuned by (grouped) grid search on the validation set, but due to the large hyperparameter space, we conduct tuning mostly for sensitive hyperparameters and admit that it is empirical. We choose the VAE as our baseline, and the KL weight of VAE is tuned in the same way. We list the hyperparameters in Appendix A.

The training objective is optimized by Adam (Kingma and Ba, 2015) with β_1 = 0.9, β_2 = 0.995, and the initial learning rate is 0.001. Word embeddings are 300-dimensional and initialized randomly. The dimension of each latent space (namely, z_{syn} and z_{sem}) is 100.

KL Annealing and Word Dropout We adopt the tricks of KL annealing and word dropout from Bowman et al. (2016) to avoid KL collapse. We anneal $\lambda_{\text{syn}}^{\text{KL}}$ and $\lambda_{\text{syn}}^{\text{KL}}$ from zero to predefined values in a sigmoid manner. Besides, the word dropout trick randomly replaces the ground-truth token with <unk> with a fixed probability of 0.50 at each time step of the decoder during training.

4 Experiments

We evaluate our method on reconstruction and unconditional language generation (§4.1). Then, we apply it two applications, namely, unsupervised paraphrase generation (§4.2) and syntax-transfer generation (§4.3).

4.1 Reconstruction and Unconditional Language Generation

First, we compare our model in reconstruction and unconditional language generation with a traditional VAE and a syntactic language model (PRPN, Shen et al., 2017).

Dataset We followed previous work (Bowman et al., 2016) and used a standard benchmark, the WSJ sections in the Penn Treebank (PTB) (Marcus et al., 1993). We also followed the standard split: Sections 2–21 for training, Section 24 for validation, and Section 23 for test.

Settings We trained VAE and DSS-VAE, both with 100-dimensional RNN states. For the vocabulary, we chose 30k most frequent words. We trained PRPN with the default parameter in the code base.[2]

Evaluation We evaluate model performance with the following metrics:

1. Reconstruction BLEU. The reconstruction task aims to generate the input sentence itself. In the task, both syntactic and semantic vectors are chosen as the predicted

[2]https://github.com/yikangshen/PRPN

KL-Weight	BLEU$^\uparrow$	Forward PPL$^\downarrow$
1.3	7.26	34.01
1.2	7.41	35.00
1.0	8.19	36.53
0.7	8.98	42.44
0.5	9.07	44.11
0.3	9.26	48.70
0.1	9.36	49.73

Table 1: BLEU and Forward PPL of VAE with varying KL weights on the PTB test set. The larger$^\uparrow$ (or lower$^\downarrow$), the better.

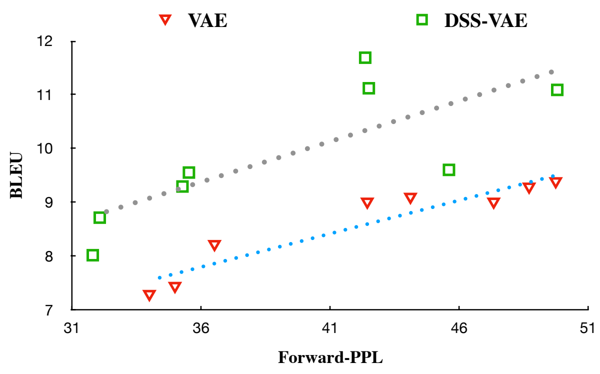

Figure 3: Comparing DSS-VAE and VAE in language generation with different KL weight. We performed linear regression for each model to show the trend. The upper-left corner (larger BLEU but smaller PPL) indicates a better performance.

mean of the encoded distribution. We evaluate the reconstruction performance by the BLEU score (Papineni et al., 2002) with input as the reference.[3] It reflects how well the model could preserve input information, and is crucial for representation learning and "goal-oriented" text generation.

2. Forward PPL. We then perform unconditioned generation, where both syntactic and semantic vectors are sampled from prior. Forward perplexity (PPL) (Zhao et al., 2018) is the generated sentences' perplexity score predicted by a pertained language model.[4] It shows the fluency of generated sentences from VAE's prior. We computed Forward PPL based on 100K sampled sentences.

3. Reverse PPL. Unconditioned generation is further evaluated by Reverse PPL (Zhao et al., 2018). It is obtained by first training a language model[5] on 100K sampled sentences from a generation model; then, Reverse PPL is the perplexity of the PTB test sets with the trained language model. Reverse PPL evaluates the diversity and fluency of sampled sentences from a language generation model. If sampled sentences are of low diversity, the language model would be trained only on similar sentences; if the sampled sentences are of low fluency, the language model would be trained on unfluent sentences. Both will lead to higher Reverse PPL. For comparing VAE and DSS-VAE, we sample latent variables from the prior, and feed them to the decoder for generation; for LSTM-LM, we first feed the start sentence token <s> to the decoder, and sample the word at each time step by predicted probabilities (i.e., forward sampling).

[3]We evaluate the corpus BLEU implemented in https://www.nltk.org/_modules/nltk/translate/bleu_score.html

[4]We used an LSTM language model trained on the One-Billion-Word Corpus (http://www.statmt.org/lm-benchmark).

[5]Tied LSTM-LM with 300 dimensions and two layers, implemented in https://github.com/pytorch/examples/tree/master/word_language_model

Results We see in Table 1 that BLEU and PPL are more or less contradictory. Usually, a smaller KL weight makes the autoencoder less "variational" but more "deterministic," leading to less fluent sampled sentences but better reconstruction. If the trade-off is not analyzed explicitly, the VAE variant could have arbitrary results based on KL-weight tuning, which is unfair.

We therefore present the scatter plot in Figure 3, showing the trend of forward PPL and BLEU scores with different KL weights. Clearly, DSS-VAE outperforms a plain VAE in BLEU if Forward PPL is controlled, and in Forward PPL if BLEU is controlled. The scatter plot shows that our proposed DSS-VAE outperforms the original counterpart in language generation with different KL weights.

In terms of Reverse PPL (Table 2), DSS-VAE also achieves better Reverse PPL than a traditional VAE. Since DSS-VAE leverages syntax to improve the sentence generation, we also include a state-of-the-art syntactic language model (PRPN-LM, Shen et al., 2017) for comparison. Results show that DSS-VAE has achieved a Reverse PPL comparable to (and slightly better than) PRPN-LM. It is also seen that explicitly modeling syntactic structures does yield better generation results—DSS-VAE and PRPN consistently

Model	Reverse PPL$^{\downarrow}$
Real data	70.76
LSTM-LM	132.46
PRPN-LM	116.67
VAE	125.86
DSS-VAE	**116.23**

Table 2: Reverse PPL reflect the diversity and fluency of sampling data, the lower$^{\downarrow}$, the better. Training on the model sampled and evaluated on the real test set. We set the same KL weight for DSS-VAE and VAE here.(KL weight=1.0)

outperform VAE and LSTM-LM in sentence generation.

We also include the Reverse PPL of the real training sentences. As expected, training a language model on real data outperforms training on sampled sentences from a generation model, showing that there is still much room for improvement for all current sentence generators.

4.2 Unsupervised Paraphrase Generation

Given an input sentence, paraphrase generation aims to synthesize a sentence that appears different from the input, but conveys the same meaning. We propose a novel approach to unsupervised paraphrase generation with DSS-VAE. Suppose a DSS-VAE is well trained according to §3.3, our approach works in the inference stage.

For a particular input sentence $\boldsymbol{x}^*$, let $q(\boldsymbol{z}_{\text{syn}}|\boldsymbol{x}^*)$ and $q(\boldsymbol{z}_{\text{sem}}|\boldsymbol{x}^*)$ be the encoded posterior distributions of the syntactic and semantic spaces, respectively. The inferred latent vectors are:

$$\boldsymbol{z}^*_{\text{sem}} = \text{argmax}_{\boldsymbol{z}_{\text{sem}}}\ q(\boldsymbol{z}_{\text{sem}}|\boldsymbol{x}^*) \tag{9}$$

$$\boldsymbol{z}^*_{\text{syn}} \sim q(\boldsymbol{z}_{\text{syn}}|\boldsymbol{x}^*) \tag{10}$$

and are further combined as:

$$\boldsymbol{z}^* = [\boldsymbol{z}^*_{\text{syn}}; \boldsymbol{z}^*_{\text{sem}}] \tag{11}$$

Finally, $\boldsymbol{z}^*$ is fed to the decoder and perform a greedy decoding for paraphrase generation.

The intuition behind is that, when generating the paraphrase, semantics should remain the same, but the syntax of a paraphrase could (and should) vary. Therefore, we sample a $\boldsymbol{z}^*_{\text{syn}}$ vector from its probabilistic distribution, while fixing $\boldsymbol{z}^*_{\text{sem}}$.

Dataset We used the established Quora dataset[6] to evaluate paraphrase generation, following previous work (Miao et al., 2019). The dataset contains 140k pairs of paraphrase sentences and 260k pairs of non-paraphrase sentences. In the standard dataset split, there are 3k and 30k held-out validation and test sets, respectively. In this experiment, we consider the unsupervised setting as Miao et al. (2019), using all non-paraphrase sentences as training samples. It is also noted that we only valid our model on the non-paraphrase held-out validation set by selecting with the lowest validation ELBO.

[6]https://www.kaggle.com/c/quora-question-pairs/data

Model	BLEU-ref$^{\uparrow}$	BLEU-ori$^{\downarrow}$
Origin Sentence†	30.49	100
VAE-SVG-eq (supervised)‡	22.90	–
VAE (unsupervised)†	9.25	27.23
CGMH†	18.85	50.18
DSS-VAE	**20.54**	52.77

Table 3: Performance of paraphrase generation. The larger$^{\uparrow}$ (or lower$^{\downarrow}$), the better. Some results are quoted from †Miao et al. (2019) and ‡Gupta et al. (2018).

Evaluation Since the test set contains a reference paraphrase for each input, it is straightforward to compute the BLEU against the reference, denoted by BLEU-ref. However, this metric alone does not model whether the generated sentence is different from the input, and thus, Miao et al. (2019) propose to measure this by computing BLEU against the original sentence (denoted as BLEU-ori), which ideally should be low. We only consider the DSS-VAE that yields a BLEU-ori lower than 55, which is empirically suggested by Miao et al. (2019) that ensures the obtained sentence is different from the original to at least a certain degree.

Results Table 3 shows the performance of unsupervised paraphrase generation. In the first row of Table 3, simply copying the original sentences yields the highest BLEU-ref, but is meaningless as it has a BLEU-ori score of 100. We see that DSS-VAE outperforms the CGMH and the original VAE in BLEU-ref. Especially, DSS-VAE achieves a closer BLEU-ref compared with supervised paraphrase methods (Gupta et al., 2018).

We admit that it is hard to present the trade-off by listing a single score for each model in the Table 3. We therefore have the scatter plot in Figure 4 to further compare these methods. As seen, the trade-off is pretty linear and less noisy compared with Figure 3. It is seen that the line of DSS-VAE is located to the upper-left of the competing methods. In other words, the plain VAE and CGMH are “inadmissible,” meaning that DSS-

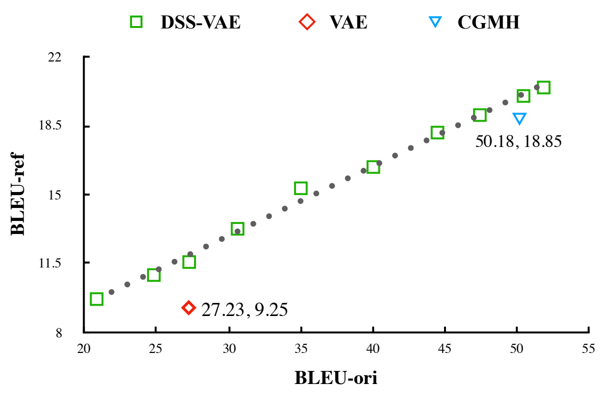

Figure 4: Trade-off between BLEU-ori (the lower, the better) and BLEU-ref (the larger, the better) in unsupervised paraphrase generation. Again, the upper-left corner indicates a better performance.

VAE simultaneously outperforms them in both BLEU-ori and BLEU-ref, indicating that DSS-VAE outperforms previous state-of-the-art methods in unsupervised paraphrase generation.

4.3 Syntax-Transfer Generation

In this experiment, we propose a novel application of syntax-transfer text generation, inspired by previous sentiment-style transfer studies (Hu et al., 2017; Fu et al., 2018; John et al., 2018).

Consider two sentences:

$\boldsymbol{x}_1$: `There is a dog behind the door.`

$\boldsymbol{x}_2$: `The child is playing in the garden.`

If we would like to generate a sentence having the syntax of "`there is/are`" as $\boldsymbol{x}_1$ but conveying the meaning of $\boldsymbol{x}_2$, we could graft the respective syntactic and semantic vectors as:

$$\begin{aligned} \boldsymbol{z}^*_{\text{sem}} &= \operatorname{argmax}_{\boldsymbol{z}_{\text{sem}}} q(\boldsymbol{z}_{\text{sem}}|\boldsymbol{x}_2) \\ \boldsymbol{z}^*_{\text{syn}} &= \operatorname{argmax}_{\boldsymbol{z}_{\text{syn}}} \quad q(\boldsymbol{z}_{\text{syn}}|\boldsymbol{x}_1) \\ \boldsymbol{z} &= \left[\boldsymbol{z}^*_{\text{sem}}; \boldsymbol{z}^*_{\text{syn}}\right] \end{aligned}$$

and then feed $\boldsymbol{z}$ to the decoder to obtain a syntax-transferred sentence.

Dataset and Evaluation To evaluate this task, we constructed a subset of the Stanford Natural Language Inference (SNLI), containing 1000 non-paraphrase pairs. SNLI sentences can be thought of as a simple domain-specific corpus, but were all written by humans. In each pair we constructed, one sentence serves as the semantic provider (denoted by Ref$_{\text{sem}}$), and the other serves as the syntactic provider (denoted by Ref$_{\text{syn}}$). The goal of syntax-transfer text generation is to synthesize a sentence that resembles Ref$_{\text{sem}}$ but not Ref$_{\text{syn}}$ in semantics, and resembles Ref$_{\text{syn}}$ but not Ref$_{\text{sem}}$ in syntax. For the semantic part, we use the traditional word-based BLEU scores to evaluate how the generated sentence is close to Ref$_{\text{sem}}$ but different from Ref$_{\text{syn}}$. For syntactic similarity, we use the zss package[7] to calculate the Tree Edit Distance (TED, Zhang and Shasha, 1989). TED is essentially the minimum-cost sequence of node edit operations (namely, delete, insert, and rename) between two trees, which reflects the difference of two syntactic trees.

Since we hope the generated sentence has a higher word-BLEU score compared with Ref$_{\text{sem}}$ but a lower word-BLEU score compared with Ref$_{\text{syn}}$, we compute their difference, denoted by Δword-BLEU, to consider both. Likewise, ΔTED is also computed. We further take the geometric mean of Δword-BLEU and ΔTED to take both into account.

Results We see from Table 4 that a traditional VAE cannot accomplish the task of syntax transfer. This is because Ref$_{\text{syn}}$ and Ref$_{\text{sem}}$—even if we artificially split the latent space into two parts—play the same role in the decoder. With the multi-task and adversarial losses for syntactic and semantic latent spaces, the total difference is increased by 12.09, which shows the success of syntax-transfer sentence generation. This further implies that explicitly modeling syntax is feasible in the latent space of VAE. We incrementally applied the adversarial reconstruction loss, proposed in § 3.2.3.

As seen, an adversarial reconstruction loss drastically strengthens the role of the other space. For example, $+\mathcal{L}^{(\text{adv})}_{\text{rec}}(\boldsymbol{z}_{\text{sem}})$ repels information to the syntactic space and achieves the highest ΔTED.

When applying the adversarial reconstruction losses to both semantic and syntactic spaces, we have a balance between Δword-BLEU and ΔTED, both ranking second in the respective columns. Eventually, we achieve the highest total difference, showing that our full DSS-VAE model achieves the best performance of syntax-transfer generation.

Discussion on syntax transfer between incompatible sentences We provide a few case studies of syntax-transfer generation in Appendix B. We empirically find that the syntactic transfer be-

[7]https://github.com/timtadh/zhang-shasha

Model	word-BLEU (corpus)		Δword-BLEU$^{\uparrow}$	Average TED (per sentence)		ΔTED$^{\uparrow}$	Geo Mean $\Delta^{\uparrow}$
	Ref$_{\text{sem}}$$^{\uparrow}$	Ref$_{\text{syn}}$$^{\downarrow}$		Ref$_{\text{sem}}$$^{\uparrow}$	Ref$_{\text{syn}}$$^{\downarrow}$		
VAE	6.81	6.68	0.13	149.22	148.59	0.63	0.29
$\mathcal{L}_{\text{sem}}^{(\text{mul})} + \mathcal{L}_{\text{syn}}^{(\text{mul})} + \mathcal{L}_{\text{sem}}^{(\text{adv})} + \mathcal{L}_{\text{syn}}^{(\text{adv})}$	12.14	6.22	5.92	159.51	134.80	24.71	12.09
$+\mathcal{L}_{\text{rec}}^{(\text{adv})}(z_{\text{sem}})$	11.83	6.60	5.23	**163.40**	**131.27**	**32.13**	12.96
$+\mathcal{L}_{\text{rec}}^{(\text{adv})}(z_{\text{syn}})$	**14.33**	**6.07**	**8.26**	159.20	134.22	24.98	14.36
$+\mathcal{L}_{\text{rec}}^{(\text{adv})}(z_{\text{syn}}) + \mathcal{L}_{\text{rec}}^{(\text{adv})}(z_{\text{sem}})$	13.74	6.15	7.59	161.94	131.09	30.85	**15.30**

Table 4: Performance of syntax-transfer generation. The larger$^{\uparrow}$ (or lower$^{\downarrow}$), the better. The results of VAE are obtained by averaging interpolation. Δword-BLEU $=$ word-BLEU(Ref$_{\text{sem}}$) $-$ word-BLEU(Ref$_{\text{syn}}$). We also compute the difference as ΔTED $=$ TED(Ref$_{\text{sem}}$) $-$ TED(Ref$_{\text{syn}}$) to measure if the generated sentence is syntactically similar to Ref$_{\text{syn}}$ but not Ref$_{\text{sem}}$. Due to the difference of scale between BLEU and TED, we compute the geometric mean of Δword-BLEU and ΔTED reflect the total differences.

tween "compatible" sentences give more promising results than transfer between "incompatible" sentences. Intuitively, this is reasonable because it may be hard to transfer a sentence with a length of 5, say, to a sentence with a length of 50.

5 Conclusion

In this paper, we propose a novel DSS-VAE model, which explicitly models syntax in the distributed latent space of VAE and enjoys the benefits of sampling and manipulation in terms of the syntax of a sentence. Experiments show that DSS-VAE outperforms the VAE baseline in reconstruction and unconditioned language generation. We further make use of the sampling and manipulation advantages of DSS-VAE in two novel applications, namely unsupervised paraphrase and syntax-transfer generation. In both experiments, DSS-VAE achieves promising results.

Acknowledgments

We would like to thank the anonymous reviewers for their insightful comments. This work is supported by the National Science Foundation of China (No. 61772261 and No. 61672277) and the Jiangsu Provincial Research Foundation for Basic Research (No. BK20170074).

References

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *CoNLL*, pages 10–21.

Eugene Charniak. 2001. Immediate-head parsing for language models. In *ACL*, pages 124–131.

Ciprian Chelba. 1997. A structured language model. In *ACL*, pages 498–500.

Huadong Chen, Shujian Huang, David Chiang, and Jiajun Chen. 2017. Improved neural machine translation with a syntax-aware encoder and decoder. In *ACL*, pages 1936–1945.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *EMNLP*, pages 1724–1734.

Alexander Clark. 2001. Unsupervised induction of stochastic context-free grammars using distributional clustering. In *Proceedings of the ACL 2001 Workshop on Computational Natural Language Learning*.

Hanjun Dai, Yingtao Tian, Bo Dai, Steven Skiena, and Le Song. 2018. Syntax-directed variational autoencoder for structured data. In *ICLR*.

Jan Milan Deriu and Mark Cieliebak. 2018. Syntactic manipulation for generating more diverse and interesting texts. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 22–34.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A Smith. 2016. Recurrent neural network grammars. In *NAACL*, pages 199–209.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016. Tree-to-sequence attentional neural machine translation. In *ACL*, pages 823–833.

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. 2018. Style transfer in text: Exploration and evaluation. In *AAAI*, pages 663–670.

Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. 2018. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 4(2):268–276.

Ankush Gupta, Arvind Agarwal, Prawaan Singh, and Piyush Rai. 2018. A deep generative framework for paraphrase generation. In *AAAI*, pages 5149–5156.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Toward controlled generation of text. In *ICML*, pages 1587–1596.

Vineet John, Lili Mou, Hareesh Bahuleyan, and Olga Vechtomova. 2018. Disentangled representation learning for text style transfer. *arXiv preprint arXiv:1808.04339*.

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

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

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig, and Noah A Smith. 2017. What do recurrent neural network grammars learn about syntax? In *EACL*, pages 1249–1258.

Matt J. Kusner, Brooks Paige, and José Miguel Hernández-Lobato. 2017. Grammar variational autoencoder. In *ICML*, pages 1945–1954.

Juncen Li, Robin Jia, He He, and Percy Liang. 2018a. Delete, Retrieve, Generate: a simple approach to sentiment and style transfer. In *ACL*, pages 1865–1874.

Junhui Li, Deyi Xiong, Zhaopeng Tu, Muhua Zhu, Min Zhang, and Guodong Zhou. 2017. Modeling source syntax for neural machine translation. In *ACL*, pages 688–697.

Juntao Li, Yan Song, Haisong Zhang, Dongmin Chen, Shuming Shi, Dongyan Zhao, and Rui Yan. 2018b. Generating classical chinese poems via conditional variational autoencoder and adversarial training. In *EMNLP*, pages 3890–3900.

Lemao Liu, Muhua Zhu, and Shuming Shi. 2018. Improving sequence-to-sequence constituency parsing. In *AAAI*, pages 4873–4880.

Shuming Ma, Xu Sun, Yizhong Wang, and Junyang Lin. 2018. Bag-of-words as target for neural machine translation. In *ACL*, pages 332–338.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The Penn Treebank. *Computational linguistics*, 19(2):313–330.

Ning Miao, Hao Zhou, Lili Mou, Rui Yan, and Lei Li. 2019. CGMH: Constrained sentence generation by Metropolis-Hastings sampling. In *AAAI*.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *ICML*, pages 807–814.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *ACL*, pages 311–318.

Stanislau Semeniuta, Aliaksei Severyn, and Erhardt Barth. 2017. A hybrid convolutional variational autoencoder for text generation. In *EMNLP*, pages 627–637.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI*, pages 3295–3301.

Yikang Shen, Zhouhan Lin, Chin-Wei Huang, and Aaron Courville. 2017. Neural language modeling by jointly learning syntax and lexicon. In *ICLR*.

Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. 2017. Adversarial discriminative domain adaptation. In *CVPR*, pages 7167–7176.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*, pages 5998–6008.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *NIPS*, pages 2773–2781.

Rongxiang Weng, Shujian Huang, Zaixiang Zheng, XIN-YU DAI, and CHEN Jiajun. 2017. Neural machine translation with word predictions. In *EMNLP*, pages 136–145.

Kaizhong Zhang and Dennis Shasha. 1989. Simple fast algorithms for the editing distance between trees and related problems. *SIAM journal on computing*, 18(6):1245–1262.

Junbo Jake Zhao, Yoon Kim, Kelly Zhang, Alexander M. Rush, and Yann LeCun. 2018. Adversarially regularized autoencoders. In *ICML*, pages 5897–5906.

Hao Zhou, Zhaopeng Tu, Shujian Huang, Xiaohua Liu, Hang Li, and Jiajun Chen. 2017. Chunk-based bi-scale decoder for neural machine translation. In *ACL*, pages 580–586.

A Hyperparameter Details

We list the hyperparaemters in Tables 5 and 6. Every 500 batch, we save the model if it achieves a lower *evidence lower bound* (ELBO) on the validation set.

Hyper-parameters	Value
$\lambda_{sem}^{\mathrm{KL}}$	1.0
$\lambda_{syn}^{\mathrm{KL}}$	1.0
$\lambda_{sem}^{\mathrm{mul}}$	0.5
$\lambda_{syn}^{\mathrm{mul}}$	0.5
$\lambda_{sem}^{\mathrm{adv}}$	0.5
$\lambda_{syn}^{\mathrm{adv}}$	0.5
$\lambda_{sem}^{\mathrm{rec}}$	0.5
$\lambda_{syn}^{\mathrm{rec}}$	0.5
Batch size	32
GRU Dropout	0.1

Table 5: The hyper-parameters we used in PTB dataset

Hyper-parameters	Value
$\lambda_{sem}^{\mathrm{KL}}$	1/3
$\lambda_{syn}^{\mathrm{KL}}$	2/3
$\lambda_{sem}^{\mathrm{mul}}$	5.0
$\lambda_{syn}^{\mathrm{mul}}$	1.0
$\lambda_{sem}^{\mathrm{adv}}$	0.5
$\lambda_{syn}^{\mathrm{adv}}$	0.5
$\lambda_{sem}^{\mathrm{rec}}$	1.0
$\lambda_{syn}^{\mathrm{rec}}$	0.05
Batch size	50
GRU Dropout	0.3

Table 6: The hyper-parameters we used in Quora dataset.

Semantic and Syntactic Providers		Syntax-Transfer Output	
Ref$_{\mathrm{syn}}$:	There is an apple on the table.	**VAE**:	The man is in the kitchen.
Ref$_{\mathrm{sem}}$:	The airplane is in the sky.	**DSS-VAE**:	There is a airplane in the sky.
Ref$_{\mathrm{syn}}$:	The shellfish was cooked in a wok.	**VAE**:	The man was filled with people.
Ref$_{\mathrm{sem}}$:	The stadium was packed with people.	**DSS-VAE**:	The stadium was packed with people.
Ref$_{\mathrm{syn}}$:	The child is playing in the garden.	**VAE**:	There is a person in the garden.
Ref$_{\mathrm{sem}}$:	There is a dog behind the door.	**DSS-VAE**:	A dog is walking behind the door.

Table 7: Case studies of syntax transfer generation.

B Case Study of Syntax Transfer

We provide a few examples in Table 7. We see in all cases that a plain VAE "interpolates" two sentences without the consideration of syntax and semantics, whereas our DSS-VAE is able to transfer the syntax without changing the meaning much. In the first example, DSS-VAE successfully transfer a "subject-be-predicative" sentence to a "there is/are" sentence. For the second example, the semantic reference has the same syntactic structure as the syntax reference, and as a result, DSS-VAE generates the same sentence as Ref$_{\mathrm{sem}}$. For the last example, we transfer a "there is/are" sentence to a "subject-be-predicative" sentence, and our DSS-VAE is also able to generate the desired syntax.