Multi-type Disentanglement without Adversarial Training

Lei Sha, Thomas Lukasiewicz
Department of Computer Science, University of Oxford, UK
{lei.sha, thomas.lukasiewicz} @cs.ox.ac.uk

Abstract

Controlling the style of natural language by disentangling the latent space is an important step towards interpretable machine learning. After the latent space is disentangled, the style of a sentence can be transformed by tuning the style representation without affecting other features of the sentence. Previous works usually use adversarial training to guarantee that disentangled vectors do not affect each other. However, adversarial methods are difficult to train. Especially when there are multiple features (e.g., sentiment, or tense, which we call style types in this paper), each feature requires a separate discriminator for extracting a disentangled style vector corresponding to that feature. In this paper, we propose a unified distribution-controlling method, which provides each specific style value (the value of style types, e.g., positive sentiment, or past tense) with a unique representation. This method contributes a solid theoretical basis to avoid adversarial training in multi-type disentanglement. We also propose multiple loss functions to achieve a style-content disentanglement as well as a disentanglement among multiple style types. In addition, we observe that if two different style types always have some specific style values that occur together in the dataset, they will affect each other when transferring the style values. We call this phenomenon training bias , and we propose a loss function to alleviate such training bias while disentangling multiple types. We conduct experiments on two datasets (Yelp service reviews and Amazon product reviews) to evaluate the style-disentangling effect and the unsupervised style-transfer performance on two style types: sentiment and tense. The experimental results show the effectiveness of our model.

1 Introduction

Changing some specific features (such as sentiment, tense, or human face pose) of a given text or image is very important in many applications. These features are usually embedded in the weights of "black-box" deep neural networks. Hence, disentangling the latent space of the neural network becomes a valuable step of such tasks.

In the early years, researchers tried to use some disentangled latent variables to control latent features of images and text. For example, Chen et al. (2016) use scalar latent variables to control the writing style for handwritten digits as well as the pose of human face 3D-rendered images by maximizing the mutual information between the latent variable and the generator. Furthermore, there is previous research that tends to completely separate the latent space into disentangled components. For example, John et al. (2019) use multiple adversarial optimizers to decrease the dependency between the latent vectors of style and content.

A severe limitation in previous works is adversarial training, which is always difficult to train and usually requires many resources. Especially when we are extracting multiple style vectors and each of them represents a specific feature, then for each kind of feature, there needs to be a discriminator, according to John et al. (2019). Therefore, a generator with so many discriminators will make the training process extremely complicated.

In this paper, we distinguish the concepts of style type and value: (1) a *style type* is a style class that represents a specific feature of text or an image, e.g., sentiment, tense, or face direction; and (2) a *style value* is one of the different values within a style type, e.g., sentiment (positive/negative), or tense (past/now/future). We propose a unified distribution-controlling method that gives a unique representation to each style value in each style type. We assume that the representations of each style value are sampled from separate Gaussian distributions. Then, we generate multiple style vectors (for different style types) and a content vector using the input text, and force the style vectors to be close to the ground-truth style-value distributions. To ensure that the semantic information is not lost, we sample a style vector from the ground-truth style-value distribution for each style type, and combine all the style vectors and the content vector into one vector. Then, we force the combined vector to reconstruct the original sentence. To avoid adversarial training, we propose a loss function for style-content disentanglement to make it more efficient, which is applicable to both vanilla and variational autoencoders. We further propose loss functions for the disentanglement among multiple style types. In addition, we point out a severe problem in multi-type disentanglement, which is called *training bias*. We prove mathematically that our multi-type disentanglement loss function can alleviate the training bias problem.

We conducted experiments on two datasets: Yelp Service Reviews (Shen et al. 2017) and Amazon Product Reviews (Fu et al. 2018). The experimental results show that our method can provide a comparable disentanglement effect and even better style-transfer effect without the help of adversarial training. The experimental results also show our method's effectiveness for alleviating the training bias.

The contributions of this paper are briefly as follows:

- We propose a unified distribution-controlling method for disentanglement, which provides unique representations for each style value in each style type and provides a natural advantage for multi-type disentanglement.
- We propose loss functions to disentangle style and content without adversarial training. Our method is applicable even in the situation where one type contains multiple style values, both in vanilla and variational autoencoders.
- We propose loss functions for disentangling multiple style types, which can also alleviate the bias caused by multi-type training data. Based on a solid theory, the effectiveness of these loss functions is also shown in experiments.

The rest of this paper is organized as follows. Section 2 introduces the details of our approach and the losses designed for style-content and multi-type disentanglement. We then conduct experiments in Section 3, discuss related work in Section 4, and finally conclude our work in Section 5.

2 Approach

In this section, we first make an assumption about style vectors, called *unified distribution assumption*. On top of this, we then build our model architecture in Fig. 1. We further propose two loss functions for style-content disentanglement (which encourage that the style vectors and the content vector do not affect each other) and multi-type disentanglement (which reduces the effect among the style vectors).

2.1 Unified Distribution-Controlling Method

Intuitively, for the sentences belonging to one specific style value, the style vector generated by them should have the same representation. Since this is a very strong condition, we use a relaxed form of this requirement as *unified distribution assumption*, in which we require that all the vectors that belong to one specific style value follow a unified distribution, the parameters of which can also be updated during training. We use a Gaussian distribution with different mean and variance for each style value. Then, we require the disentangled vectors to satisfy the following requirements.

- Disentangled vectors that belong to the same style value should obey the same Gaussian distribution.
- The Gaussian distributions corresponding to any two different style values should be independent from each other.

The control under the unified distribution assumption is as follows. When we are conducting style transfer, we only need to sample a new style vector from the target style value and replace the original one, then the corresponding style of the generated sentence will also be transferred.

2.2 Model

We now describe our model in detail. As shown in Fig. 1, the basic architecture of our method is an autoencoder. We first encode the input sentence and generate several disentangled style vectors and a content vector from the encoded sentence. The style vector is required to be close to the correct unified style distribution. Ideally, after the sentence is perfectly disentangled into style and content vector, if the style is replaced by the unified representation of the same style value, the original sentence can be correctly reconstructed by the new style vector and the original content vector. Therefore, in the final step, we sample a vector from the correct unified style distribution to replace the original one and reconstruct the original sentence.

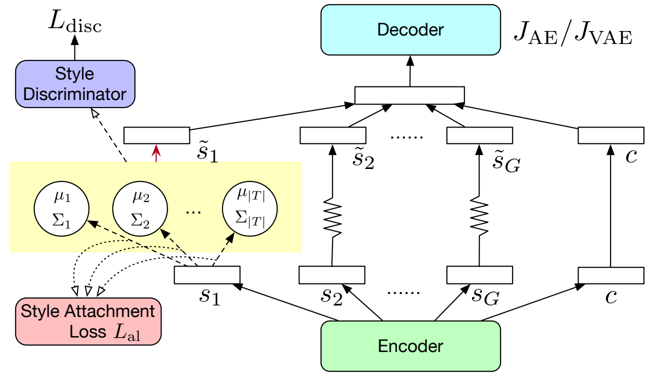

Figure 1: Architecture of our method: the circles labeled with μ_i and Σ_i represent the style-value distributions, while the zig-zag arrows are the omission of the sampling process from the correct style value.

Autoencoder. In disentanglement approaches, the best way to prevent the loss of information is using autoencoders, which encode the input sentence to a latent space and then reconstruct the original sentence from this space. In our method, we use two kinds of autoencoders: a vanilla and a variational autoencoder (Kingma and Welling 2014). Each training example is composed of a sentence x and a bunch of style values corresponds to different style types: $t_{1\star}, \ldots, t_{G\star}$[1] ($G$ is the number of style types). Given the input token sequence $x = \{x_1, x_2, \ldots, x_n\}$, we use an LSTM-based (Hochreiter and Schmidhuber 1997) vanilla and a variational autoencoder to build the reconstruction loss:

$$\begin{aligned} J_{\text{AE}} &= -\sum_t \log P(x_t|h, x_1, x_2, \ldots, x_{t-1}) \\ J_{\text{VAE}} &= -\int q_E(h|x) \log[P(x|h)]dh \\ &\quad + \lambda_{\text{KL}} \text{KL}(q_E(h|x)||P(h)), \end{aligned} \tag{1}$$

where h is the latent vector, $P(x|h)$ is the decoder part, $P(h)$ is the prior distribution of h (usually, $\mathcal{N}(0, 1)$), and λ_{KL} is a hyperparameter. The latent vector is then split into G style vectors $s_1, \ldots, s_G$ of equal length and a content vector c.

Style Attachment Loss on Latent Style Space. For any style type, we need to make the style vector $s_\star$ "appear like"

[1] We use t_{ij} to represent the j-th value of the i-th style type, and "$\star$" means that we have the same action for each style type (when it replaces i) or value (when it replaces j).

sampled from the correct style value's unified distribution. For simplicity, we omit the subscripts and use s for style vector, and t for style value in this section. So, we need to maximize the probability of the style vector s belonging to style value t, denoted $P(t|s)$.

In Fig. 1, the Gaussian distributions of the style values have parameters $\mu_{\star j}, \Sigma_{\star j}, j \in \{1, \ldots, |T|\}$ (T represents the set of style values, $|T|$ represents the number of style values). Then, the probability density function (PDF) of the j-th style value $T_{\star j}$ is as follows:

$$p_{\text{Nor}}(s|T_{\star j}) = \frac{\exp\Big(-\frac{1}{2}(s-\mu_{\star j})\Sigma_{\star j}^{-1}(s-\mu_{\star j})\Big)}{\sqrt{(2\pi)^d \det(\Sigma_{\star j})}}, \tag{2}$$

where d means the dimension of the style vector, and "Nor" is short for "Normal distribution". Then, we use Bayes' theorem to calculate this probability as shown in Eq. 3:

$$P(t|s) = \frac{p_{\text{Nor}}(s|t)p(t)}{p(s)} = \frac{p_{\text{Nor}}(s|t)p(t)}{\sum_{t' \in T}(p_{\text{Nor}}(s|t')p(t'))}, \tag{3}$$

where $p(t)$ is the prior distribution of the style values, which is decided by the dataset.

Therefore, we define the style attachment loss as the negative log-likelihood (NLL) loss according to the labeled style value:

$$L_{\text{al}} = -\frac{1}{|D|}\sum_{m=1}^{|D|} \log P(t^{(m)}|s^{(m)}), \tag{4}$$

where $|D|$ denotes the size of the training set, and $t^{(m)}$ represents the label of the m-th case in the training set.

Style Classification Loss. We need to make each unified distribution really map to the corresponding style value, so we sample vectors from each style value distribution and force them to be classified to that style value.

We still use the Gaussian distribution to calculate the classification loss. We first sample M vectors from the distributions $\mathcal{N}(\mu_{\star j}, \Sigma_{\star j})$, denoted $\tilde{s}_{\star j}^{(m)}$ (which is the m-th sample from the j-th style value distribution). Then, we calculate the probability of $\tilde{s}_{\star j}^{(m)}$ for the style value $T_{\star j}$ as in Eq. 5:

$$P_{\text{c}}(T_{\star j}|\tilde{s}_{\star j}^{(m)}) = \frac{p_{\text{Nor}}(\tilde{s}_{\star j}^{(m)}|T_{\star j})}{\sum_{t' \in T} p_{\text{Nor}}(\tilde{s}_{\star j}^{(m)}|t')}. \tag{5}$$

Since the distribution's parameters $\mu_{\star j}$ and $\Sigma_{\star j}$ also need to be updated in the training phase, we use a reparameterization trick (Devroye 1996; Kingma and Welling 2014) to make the sampling process differentiable:

$$\epsilon \sim \mathcal{N}(\mathbf{O}, \mathbf{I}), \quad \tilde{s}_{\star j}^{(m)} = A_{\star j}\epsilon + \mu_{\star j}, \tag{6}$$

where $\mathbf{O}$ is an all-zero matrix, $\mathbf{I}$ is an identity matrix, and $A_{\star j}A_{\star j}^{\top} = \Sigma_{\star j}$. So, we take $A_{\star j}$, $j \in \{1, 2, \ldots, |T|\}$, as the distribution parameters that need to be trained instead of $\Sigma_{\star j}$. Then, we define the classification loss as an NLL loss:

$$L_{\text{cl}} = -\frac{1}{M|T|}\sum_{j=1}^{|T|}\sum_{m=1}^{M} \log P_{\text{c}}(T_{\star j}|\tilde{s}_{\star j}^{(m)}). \tag{7}$$

2.3 Style-Content Disentanglement

We need to guarantee that the content vector does not contain anything about the style. Since we need to disentangle the content vector with both the style vectors before and after the style value sampling, we propose to minimize two pieces of mutual information: $I(c, t)$ between the content vector c and the style labels t, and $I(c, s)$ between the content vector c and the style vectors s.

I(c,t): To minimize the mutual information $I(c, t)$, we only need to minimize its upper bound, which can be stated as follows (a detailed proof is given in the extended paper):

$$\begin{aligned} I(c,t) &= \mathbb{E}_x\Big[\int_c \sum_t p(c,t|x)\log\tfrac{p(c,t|x)}{p(c|x)p(t)}\Big] \\ &\leq \mathbb{E}_x\Big[\sum_{t'} p(t')KL(p(c|t,x)||p(c|t',x))\Big], \end{aligned} \tag{8}$$

where $p(t')$ is a constant in specific datasets, $p(c|t,x)$ need to be modeled by another Gaussian distribution $\mathcal{N}_c(\mu'_t, \Sigma'_t)$, and then we force all the content vectors with label t to obey this distribution. To achieve that, we minimize the negative log-likelihood of the style labels in a batch.

$$L_{\text{prob-nll}} = -\frac{1}{M}\sum_{m=1}^{M} \log p_c(c^{(m)}|t^{(m)}), \tag{9}$$

where $p_c(c^{(m)}|t^{(m)})$ is obtained from the Gaussian distribution $\mathcal{N}_c(\mu'_t, \Sigma'_t)$ in a similar way as $p_{\text{Nor}}(s|T_{\star j})$ in Eq. 2, and M represents the batch size.

So, the loss function of style-content disentanglement is shown as follows.

$$\begin{aligned} L_{\text{sc}} = \mathbb{E}_x\Big[&\sum_{t'} p(t')KL(p_c(c|t,x)||p_c(c|t',x))\Big] \\ &+ \lambda_{\text{sc}} L_{\text{prob-nll}}, \end{aligned} \tag{10}$$

where λ_{sc} is a predefined hyperparameter, and $p_c(c|t)$ is constrained by the loss in Eq. 9. The KL divergence item is tractable, because p_c is a Gaussian distribution.

According to the final form of L_{sc}, our method is similar to the previously proposed variational fair autoencoder (Louizos et al. 2015). In their method, they propose a maximum mean discrepancy penalty as a regularizer to the model, which encourages the statistical moments of two classes to be the same. Differently from them, we build a prior distribution to each label and encourage these distributions to be the same, which is easy to apply to multi-class cases. On the other hand, the method of compressing the input x out of c (Moyer et al. 2018) is more eligible on variational autoencoders. In comparison, our derivation of $I(c, t)$ is based on our Gaussian assumption, which is eligible for both vanilla and variational autoencoder architectures.

I(c,s): We prove (in the extended paper) that minimizing $I(c, s)$ is equal to minimizing $KL(p(c|x)||p(c))$ and $KL(p(s|x)||p(s))$, which is just the regularization term in variational autoencoders. So, we do not have any loss function for $I(c, s)$.

2.4 Multi-type Disentanglement

There are always multiple style types occurring in a text or image, such as sentiment polarity and tense. We tend to solve two problems in this section: (1) make the style vectors $s_1, \dots, s_G$ (G is the number of style types) for different style types independent to each other, and (2) when the training set has labels of multiple types, there will be a *training bias* that makes different types affect each other. We can write the training bias as $p(T_{i\star}|T_{j\star}) > p(T_{i\star}), i \neq j$, where $T_{i\star}$ and $T_{j\star}$ stand for style value labels in style types i and j, respectively.[2] For example, if positive sentiment always occurs together with the past tense, then the positive sentiment style vector tends to carry information of the past tense.

Implicit disentanglement (Higgins et al. 2017; Chen et al. 2018) uses unsupervised methods for scalar multi-type disentanglement, where each dimension of the latent vector encodes a specific feature (style type). Inspired by β-TC-VAEs (Chen et al. 2018), multi-type vector disentanglement can also be done by minimizing the *total correlation* term:

$$KL(q(s_1, \dots, s_G)|| \textstyle\prod_i q(s_i)), \tag{11}$$

where G means the number of different types.

In our unified distribution settings, the s_i's are generated by their own style-value distribution instead of the input x. So, $q(s_1, s_2 \dots, s_G)$ can be factorized as Eq. 12 ($T_{1x}, \dots, T_{Gx}$ are the corresponding style values of x in G types):

$$\begin{aligned} q(s_1, s_2 \dots, s_G) &= \mathbb{E}_x[\textstyle\prod_i q(s_i|T_{1x}, \dots, T_{Gx})] \\ &= \textstyle\sum_x \left[\frac{\prod_i^G q(T_{1x}, \dots, T_{Gx}|s_i) q(s_i)}{p(x)^{G-1}}\right]. \end{aligned} \tag{12}$$

The proof is shown in the extended paper. Usually, since T_{ix} are potentially related, we have:

$$q(T_{1x}, \dots, T_{Gx}|s_i) = \textstyle\prod_j^G q(T_{jx}|s_i, T_{kx(k=1\dots G, k\neq j)}). \tag{13}$$

Then, we have the following theorem (proved in the extended paper) to achieve multi-type disentanglement. Here, $\mathcal{H}(\cdot)$ denotes the entropy of a probability distribution.

Theorem 2.1. *For random vector variables* $s_1, \dots, s_G$ *and values* $t_1, \dots, t_G$ *sampled from G style types*[3], *if* $\mathcal{H}(p(t_i|s_i)) = 0$, *for all i, and* $\mathcal{H}(p(t_j|s_i))$ *= MAX,*[4] *for all* i, j *with* $j \neq i$, *then for all* i, j *with* $j \neq i$, *it holds that* $p(s_j|t_i) = p(s_j)$ *and* $p(t_i|t_j, s_i) = p(t_i|s_i)$.

To alleviate the training bias, we also need to ensure that $p(t_i|t_j) = p(t_i)$ instead of just $p(t_i|t_j, s_i) = p(t_i|s_i)$. So, we also need to make $\mathcal{H}(p(t_j|t_i))$, for all i, j with $j \neq i$, reach the maximum value.

According to Theorem 2.1, when we guarantee that $\mathcal{H}(p(t_i|s_i)) = 0$, for all i, and $\mathcal{H}(p(t_j|s_i)) = \text{MAX}$ and $\mathcal{H}(p(t_j|t_i)) = \text{MAX}$, both for all i, j with $j \neq i$, we can make $q(t_{1x}, \dots, t_{Gx}|s_i) = \prod_j^G q(t_{jx}|s_i)$. Then, the *total correlation* term reaches its minimum value 0 (proved in the extended paper).

[2] Ideally, it should be $p(T_{i\star}|T_{j\star}) = p(T_{i\star}), i \neq j$.

[3] For clarity, t_i is a random variable, T_{ix} is a constant label.

[4] We use "= MAX" to represent "reaches the maximum value".

Apparently, we have the loss function for multi-type disentanglement:

$$L_m = \textstyle\sum_i \sum_{j, j\neq i} \Big[\mathcal{H}(p(t_i|s_i)) - \mathcal{H}(p(t_j|s_i)) - \mathcal{H}(p(t_j|\tilde{s}_i))\Big], \tag{14}$$

where $\tilde{s}_i$ is a sample from the distribution of style value t_i.

2.5 Training and Inference

In the training phase, we minimize the following objective:

$$J = J_{\text{AE}} + \lambda_a L_{\text{al}} + \lambda_c L_{\text{cl}} + \lambda_s L_{\text{sc}} + \lambda_m L_m. \tag{15}$$

The item J_{AE} can be replaced by J_{VAE} for variational autoencoder architectures. $\lambda_a, \lambda_c, \lambda_s$, and λ_m in Eq. 15 are predefined hyperparameters.

In the inference phase, if we would like to change the current style value to another style value (within type i), we need to follow subsequent steps. First, encode the sentence to a latent representation h, and split h to obtain $s_1, \dots, s_G$ and c. Second, we sample a new style vector $\tilde{s}_i$ from the target style distribution. Finally, replace s_i with $\tilde{s}_i$ and generate a sentence in the target style by the decoder.

3 Experiments

In this section, we answer the following three questions: (1) Is the disentanglement effect comparable to previous works? (2) Is the style-transfer performance comparable to or does it even outperform previous works? (3) Will the proposed method alleviate the training bias problem?

3.1 Data and Preprocessing

We tested our method on two datasets: Yelp Service Reviews[5] (Shen et al. 2017; Zhao et al. 2018) and Amazon Product Reviews[6] (Fu et al. 2018). These datasets all contain different sentiment values (positive and negative). We annotated the tense label [7] in the Amazon dataset by matching the sentence tokens with a time word corpus, which is collected from the TimeBank dataset[8] (Pustejovsky et al. 2003). This is consistent with the method described in Hu et al. (2017). We do not label the Yelp dataset in the same way, because the time information is so vague that our automatic annotation will cause too many errors and mislead our research.

3.2 Disentanglement Effect

We use two metrics to evaluate the disentanglement effect. In the first metric, we train separate logistic regression classifiers for all the generated style vectors and content vectors. We compare the performance with previous work in Table 1.

According to Table 1, we found that the high accuracies of style vectors on Yelp and Amazon sentiment/tense are nearly identical to the corresponding accuracy of competent space (style and content), which means that the style vectors contain nearly all sentiment/tense information. Comparatively,

[5] https://github.com/shentianxiao/language-style-transfer

[6] https://github.com/fuzhenxin/textstyletransferdata

[7] The tense label files are available at: https://drive.google.com/drive/folders/1I1hOZChFPFc2LYreWp1W6l4WpdS0pnVK?usp=sharing

[8] http://timeml.org

		Yelp		Amazon		
		John et al. (2019)	Our method	John et al. (2019)	Our method	
Style Type		Sentiment		Sentiment		Tense
Random Guess		61.30		50.82		52.30
Vanilla	Style$^{\uparrow}$	**97.40**	97.31	82.10	**83.15**	92.50
	Content$^{\downarrow}$	65.80	**65.48**	67.50	**52.30**	64.40
	Style + Content$^{\uparrow}$	**97.40**	97.31	81.90	**83.15**	92.50
VAE	Style$^{\uparrow}$	**97.40**	97.37	81.00	**81.75**	92.40
	Content$^{\downarrow}$	69.70	**63.02**	69.30	**52.30**	64.40
	Style + Content$^{\uparrow}$	**97.40**	97.38	81.00	**81.75**	92.40

Table 1: The classification accuracies on each space. The up arrow means a good result should have a larger value, while the down arrow means lower is better. Note that the accuracy for content space is the lower the better, because the goal of disentanglement is to let the content space not contain any information of style. For the tense style, there are no such evaluations in previous work, so we did not list any baseline results in this table.

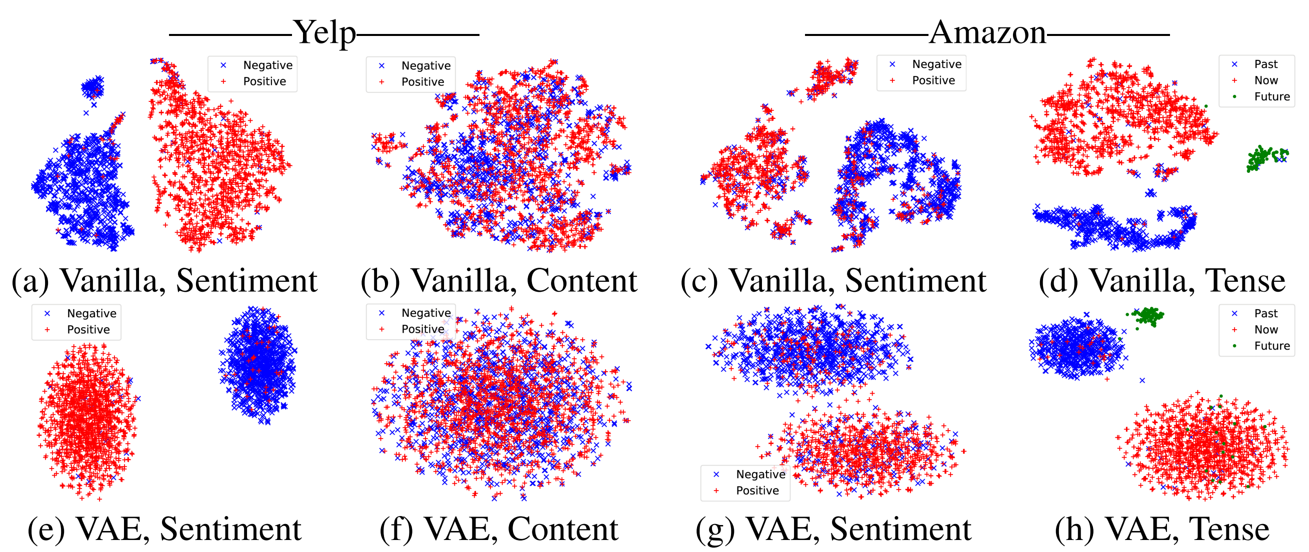

Figure 2: The t-SNE visualization of each latent space. (a), (b), (e), and (f) are the style space ("s") and content space ("c") of the Yelp dataset. (c), (d), (g), and (h) are the two style spaces (sentiment "s_1" and tense "s_2") of the Amazon dataset, which is generated simultaneously in multi-type disentanglement. All latent spaces are generated by two different methods: a vanilla autoencoder and a variational autoencoder.

the performance of content vectors are only comparable to a random guess result. These results indicate that our disentanglement method is very effective.

In Fig. 2, we listed the visualization result of each latent space using t-SNE[9] (Maaten and Hinton 2008). For the vanilla auto-encoders, we can see that in (a), (c), and (d), the different style values, such as positive/negative of sentiment and past/now/future of tense, apparently stay in separate places of themselves. The margins between any two style values are sufficient for discrimination. In Fig. 2 (b), the content space is indistinguishable due to its lack of sentiment information. For the variational autoencoders, in Fig. 2 (e), (f), (g), and (h), we found that all of them are in very regular shape. All disentangled groups of style values, including positive/negative of sentiment and past/now/future of tense, appear as separate ellipses. The reason for this phenomenon is that we force the KL divergence of each style value to be smaller, but at the same time, we force to discriminate different style values. Therefore, different style values tend to stay closer but with a large margin between them.

3.3 Style-Transfer Performance

To be consistent with previous works, we use four evaluation metrics to evaluate the style-transfer effect of our disentanglement method, and list all the results in Table 2.

(1) Style-transfer accuracy (STA): we trained two external sentence classifiers for sentiment and tense using TextCNN (Kim 2014) following previous works (Hu et al. 2017; Fu et al. 2018; John et al. 2019), and then used them to measure the sentiment/tense accuracy for the style-transferred sentences with the target style value as ground-truth. The external classifier achieves an acceptable accuracy

[9] In practice, we use the multi-core version for acceleration: https://github.com/DmitryUlyanov/Multicore-TSNE

on the validation set (97.68% on Yelp, 82.3% on Amazon Sentiment, and 96.5% on Amazon Tense), which provides a reliable approximation for the sentiment/tense accuracy.

(2) Cosine-similarity (CS): We computed the cos-similarity between the original sentence's vector and the style-transferred sentence's vector. Each sentence vector is obtained by concatenating the *max*, *min*, *mean* of word vectors after deleting the sentiment/tense words. This is again consistent with previous works (Fu et al. 2018; John et al. 2019). The goal of this metric is to evaluate the semantic similarity between the original sentence and the style-transferred sentence apart from sentiment words.

(3) Word overlap (WO): Following John et al. (2019), we calculated the unigram overlap between the original and the style-transferred sentence, which is defined as the ratio between the number of words in the intersection set and the number of words in the union set of the two sentences.

(4) Perplexity (PPL): We applied a trigram Kneser-Ney (Kneser and Ney 1995) language model as the perplexity evaluator. We trained two language models separately on respective datasets, and use the trained model to evaluate the fluency of generated sentences. A lower PPL value represents a more fluent sentence.

(5) BLEU: We calculate the BLEU 1$\sim$4 score between the original sentence and the style-transferred sentence, and take the average of BLEU 1$\sim$4 as the BLEU score. Again, we need to delete the sentiment/tense words before evaluation.

By Table 2, in Yelp and Amazon sentiment, our method is comparable with previous works on the CS and WO metrics except for (Li et al. 2018). But our variational autoencoder (VAE) architectures can outperform (Li et al. 2018) by a large margin on the STA metric. Also, our VAE architectures can outperform all the previous methods in STA on the two datasets (with Student's t-test, $p < 0.01$). In the Amazon tense experiment, although we do not have previous works to compare with, we achieved a high STA value, while the CS and WO remained comparable with the same metric in Amazon sentiment. We found that the CS and WO value of vanilla autoencoders (AEs) are higher than the same metrics of VAEs, because VAEs are more flexible in the latent space, so that the generated sentence is much more varied than vanilla AEs.

For ablation tests, we remove L_{cl} and L_{sc} from the objective for comparison. We did not try to remove L_{al}, because the whole model would become disconnected by doing that. According to Table 2, after L_{cl} is removed, the value of STA decreased a lot, because the model can no longer discriminate different style values without L_{cl}. On the other hand, when we remove L_{sc}, the STA value gets even lower. This is likely because the content space is flooded with style information without style-content disentanglement; thus, the sentence cannot be completely transferred to the target style.

3.4 Alleviation of Training Bias

When the training bias is alleviated, the value of a style type tends not to be affected by the transfer of another style type's value. So, we are trying to measure the style preservation for a baseline style type (that is held constant) while trying to transfer the other style type. Therefore, a lower deviation of STA from the original style type is better, and removing the loss L_m should produce poorer results.

The effect of our multi-type loss function on alleviating training bias is shown in Table 3, where we listed the model's accuracies on one style type when the other style type is transferred (STA_{keep} score). We report the STA_{keep} of vanilla AEs and VAEs on the two types (sentiment and tense), and we remove the loss L_m from the models for comparison. We found that the STA_{keep} score of our full model can be very close to the original sentence's STA score. But if we remove the L_m item, the accuracy of any type would decrease a lot after the style value of another type is transferred. This fact illustrates the effectiveness of our multi-type disentanglement method.

3.5 Human Evaluation

We conducted a human evaluation on the Yelp dataset and the Amazon dataset, like previous works. We randomly sampled 1,000 cases from the sentences generated by each model and asked 6 data graders to give each case a sentiment label or tense label (for transfer accuracy (TA)) and two scores on content preservation (CP), and language quality (LQ). Each score is between 1 to 5. The detailed annotation principles are listed in the extended paper. We randomly shuffled the generated sentences to conduct the grading process in a strictly blind fashion. The human evaluation results are shown in Table 4. Our measure of inter-rater agreements (the Krippendorff's alpha values (2004)) are also listed in Table 4, all of them are acceptable due to Krippendorff's principle (2004).

4 Related Work

Disentanglement is a very important method for interpretable deep learning models (Chen et al. 2019; Sha et al. 2020; Sha, Camburu, and Lukasiewicz 2021). Disentanglement works can be split into implicit disentanglement and explicit disentanglement. We summarize their characteristics as follows, focusing on the sentiment and the tense style for this comparison with previous works.

Implicit disentanglement means to disentangle meaningful factors from a variational space, but we are not sure how many disentangled components the latent space will be separated. β-VAEs (Higgins et al. 2017) and β-TCVAE (Chen et al. 2018) are unsupervised methods that extend variational autoencoders (VAEs) (Kingma and Welling 2014; Rezende, Mohamed, and Wierstra 2014) and learn disentangled representations by putting a penalty on the *total correlation (TC)* item. There are also a number of methods that extend β-VAEs and analyze different variants under specific problems (Moyer et al. 2018; Mathieu et al. 2018; Kumar, Sattigeri, and Balakrishnan 2017; Esmaeili et al. 2018; Hoffman and Johnson 2016; Narayanaswamy et al. 2017; Kim and Mnih 2018; Rezende and Viola 2018; Shao et al. 2020). However, in the training process of implicit disentanglement methods, some components may be pruned (Stühmer, Turner, and Nowozin 2019), which may lead to incorrect interpretations of the data.

In comparison, an explicit disentanglement is able to separate the latent space into more interpretable components

	Yelp					Amazon					
	STA	CS	WO	PPL	BLEU	STA/Sentiment	STA/Tense	CS	WO	PPL	BLEU
Zhao et al. (2018)	0.818	0.883	0.272	85	-	0.552	-	0.926	0.169	75	-
Li et al. (2018)	0.862	**0.941**	0.522	70	-	0.430	-	**0.976**	**0.799**	65	-
Xu et al. (2018)	0.803	0.924	0.427	470	-	0.723	-	0.912	0.222	332	-
Logeswaran et al. (2018)	0.905	0.879	0.503	133	17.4	0.857	0.942	0.908	0.356	187	16.6
Lample et al. (2019)	0.877	0.856	0.459	48	14.6	0.896	0.965	0.897	0.372	92	18.7
John et al. (2019) (Vanilla)	0.883	0.915	0.549	52	18.7	0.720	0.926	0.921	0.354	73	16.5
John et al. (2019) (VAE)	0.934	0.904	0.473	32	17.9	0.822	0.945	0.900	0.196	63	9.8
Ours (Vanilla)	0.877	0.908	**0.554**	45	16.1	0.789	0.963	**0.930**	**0.387**	68	15.4
Ours (Vanilla)$-L_{\text{cl}}$	0.605	0.852	0.419	51	13.6	0.679	0.908	0.896	0.382	74	12.5
Ours (Vanilla)$-L_{\text{sc}}$	0.383	0.875	0.550	42	15.8	0.746	0.881	0.904	**0.387**	65	16.9
Ours (VAE)	**0.944**	0.912	0.455	27	**21.2**	**0.902**	**0.993**	0.900	0.338	44	**20.1**
Ours (VAE)$-L_{\text{cl}}$	0.734	0.860	0.391	32	15.8	0.751	0.932	0.907	0.310	57	13.4
Ours (VAE)$-L_{\text{sc}}$	0.656	0.868	0.438	**25**	19.8	0.720	0.877	0.898	0.345	**43**	17.8

Table 2: The overall style transfer performance. For tense style, we do not have previous works to compare with, so we only listed our own result. STA: style transfer accuracy, CS: cosine similarity w/o sentiment/tense word, WO: word overlap w/o sentiment/tense word. For the sentiment type, the transfer direction is "Neg→Pos" and "Pos→Neg". For the tense type, the transfer direction is that "Past→Now", "Now→Future", and "Future→Past". Note that the grey numbers have very low STA, which means it always fails in style transferring. So, we highlight the best values of the evaluations apart from them.

		Sentiment	Tense
Origin	STA	0.8150	0.9715
Vanilla	STA_{keep}	0.7170	0.9220
	Δ	**0.0980**	**0.0495**
Vanilla - L_m	STA_{keep}	0.7020	0.6920
	Δ	0.1130	0.2795
VAE	STA_{keep}	0.7850	0.8825
	Δ	**0.0300**	**0.0890**
VAE - L_m	STA_{keep}	0.7260	0.8220
	Δ	0.0890	0.1495

Table 3: $\textbf{STA}_{\textbf{keep}}$ stands for the current style type's STA after another type is transferred, i.e., we observe the STA of *sentiment* when we are transferring the sentence's *tense*. Also, we observe the STA of *tense* when we are transferring *sentiment*. $\Delta = \text{STA} - \text{STA}_{\text{keep}}$. The line of "Origin" represents the CNN predicted result (STA) of the original sentence.

		Sentiment			Tense		
		TA	CP	LQ	TA	CP	LQ
Yelp	Zhao et al. (2018)	75.42	3.23	3.86	-	-	-
	John et al. (2019) (Vanilla)	82.11	3.52	4.02	-	-	-
	John et al. (2019) (VAE)	85.70	3.70	4.26	-	-	-
	Ours (Vanilla)	84.28	3.69	4.32	-	-	-
	Ours (VAE)	**86.04**	**3.78**	**4.39**	-	-	-
	IRA	0.75	0.71	0.82	-	-	-
Amazon	John et al. (2019) (Vanilla)	76.35	3.01	3.65	87.34	2.97	3.96
	John et al. (2019) (VAE)	79.60	3.26	3.76	88.92	3.14	4.14
	Ours (Vanilla)	79.03	3.34	3.74	91.09	3.21	4.09
	Ours (VAE)	**83.28**	**3.52**	**4.08**	**93.45**	**3.58**	**4.23**
	IRA	0.82	0.78	0.85	0.93	0.89	0.87

Table 4: Human evaluation results on the Yelp and Amazon dataset. Here, IRA represents inter-rater agreements.

and control them using latent variables. For example, Chen et al. (2016) present a GAN-based method that maximizes the mutual information between a scalar variable and a generator. Then, the scalar variable can be taken as a controller for the style of the generated text or image. Basically, adversarial methods are always used to guarantee that different disentangled factors are independent (John et al. 2019; Romanov et al. 2019). However, adversarial methods suffered from oscillating and unstable model parameters, which makes the training hard to converge. Also, some research (Elazar and Goldberg 2018; Moyer et al. 2018) pointed out that adversarial training is not so reliable in disentanglement-invariant representations.

One of the most common applications of disentanglement is text/image style transfer. Apart from some disentangle-free approaches (Preoţiuc-Pietro and Ungar 2018; Logeswaran, Lee, and Bengio 2018; Dai et al. 2019; Lample et al. 2019; Dankers et al. 2019), there are three ways for style transfer as follows: (1) example imitating: to extract a high-level feature and force the input text/image's feature to approach the example's feature (Gatys, Ecker, and Bethge 2016); (2) scalar variable tuning: after disentangling the input into latent scalar variables, slightly tune a scalar variable larger or smaller, expecting the generated text/image would change accordingly (Chen et al. 2016; Hu et al. 2017; Kumar, Sattigeri, and Balakrishnan 2017; Malandrakis et al. 2019); and (3) vector-variable replacing: after disentangling the input sentence, sample some target style's vectors and replace the original style vector with their average (John et al. 2019). However, an averaged style vector usually means that we have to sample some examples in the target style and calculate the average of their style vector, which is inconvenient compared to our unified style representation.

5 Conclusion

In this paper, we proposed a unified distribution method as well as multiple loss functions to avoid adversarial training in the disentangling process. Our method is easy to be applied in multi-type disentanglement. we conducted style disentanglement experiments and style transfer experiments to prove the effectiveness of our method.

Acknowledgments

This work was supported by the EPSRC grant "Unlocking the Potential of AI for English Law", a JP Morgan PhD Fellowship, the Alan Turing Institute under the EPSRC grant EP/N510129/1, and the AXA Research Fund. We also acknowledge the use of Oxford's Advanced Research Computing (ARC) facility, of the EPSRC-funded Tier 2 facility JADE (EP/P020275/1), and of GPU computing support by Scan Computers International Ltd.

References

Chen, C.; Li, O.; Tao, D.; Barnett, A.; Rudin, C.; and Su, J. K. 2019. This Looks Like That: Deep Learning for Interpretable Image Recognition. In *Advances in Neural Information Processing Systems*, 8930–8941.

Chen, T. Q.; Li, X.; Grosse, R. B.; and Duvenaud, D. K. 2018. Isolating Sources of Disentanglement in Variational Autoencoders. In *Advances in Neural Information Processing Systems*, 2610–2620.

Chen, X.; Duan, Y.; Houthooft, R.; Schulman, J.; Sutskever, I.; and Abbeel, P. 2016. InfoGAN: Interpretable Representation Learning by Information Maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, 2172–2180.

Dai, N.; Liang, J.; Qiu, X.; and Huang, X. 2019. Style Transformer: Unpaired Text Style Transfer without Disentangled Latent Representation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 5997–6007. Association for Computational Linguistics.

Dankers, V.; Rei, M.; Lewis, M.; and Shutova, E. 2019. Modelling the Interplay of Metaphor and Emotion Through Multitask Learning. 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)*, 2218–2229. Association for Computational Linguistics.

Devroye, L. 1996. Random Variate Generation in One Line of Code. In *Proceedings of the Winter Simulation Conference*, 265–272. IEEE.

Elazar, Y.; and Goldberg, Y. 2018. Adversarial Removal of Demographic Attributes from Text Data. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 11–21. Association for Computational Linguistics.

Esmaeili, B.; Wu, H.; Jain, S.; Bozkurt, A.; Siddharth, N.; Paige, B.; Brooks, D. H.; Dy, J.; and van de Meent, J.-W. 2018. Structured Disentangled Representations. *arXiv preprint arXiv:1804.02086* .

Fu, Z.; Tan, X.; Peng, N.; Zhao, D.; and Rui, Y. 2018. Style Transfer in Text: Exploration and Evaluation. In *Proceedings of the 32th AAAI Conference on Artificial Intelligence*.

Gatys, L. A.; Ecker, A. S.; and Bethge, M. 2016. Image Style Transfer Using Convolutional Neural Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2414–2423.

Higgins, I.; Matthey, L.; Pal, A.; Burgess, C.; Glorot, X.; Botvinick, M.; Mohamed, S.; and Lerchner, A. 2017. Beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. *International Conference on Learning Representations* 2(5): 6.

Hochreiter, S.; and Schmidhuber, J. 1997. Long Short-Term Memory. *Neural Computation* 9(8): 1735–1780.

Hoffman, M. D.; and Johnson, M. J. 2016. ELBO Surgery: Yet Another Way to Carve Up the Variational Evidence Lower Bound. In *Proceedings of the Workshop in Advances in Approximate Bayesian Inference, NIPS*, volume 1.

Hu, Z.; Yang, Z.; Liang, X.; Salakhutdinov, R.; and Xing, E. P. 2017. Toward Controlled Generation of Text. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 1587–1596. JMLR. org.

John, V.; Mou, L.; Bahuleyan, H.; and Vechtomova, O. 2019. Disentangled Representation Learning for Non-Parallel Text Style Transfer. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 424–434. Association for Computational Linguistics.

Kim, H.; and Mnih, A. 2018. Disentangling by Factorising. *arXiv preprint arXiv:1802.05983* .

Kim, Y. 2014. Convolutional Neural Networks for Sentence Classification. *arXiv preprint arXiv:1408.5882* .

Kingma, D. P.; and Welling, M. 2014. Auto-encoding Variational Bayes. *Proceedings of the International Conference on Learning Representations* .

Kneser, R.; and Ney, H. 1995. Improved Backing-off for M-gram Language Modeling. In *1995 International Conference on Acoustics, Speech, and Signal Processing*, volume 1, 181–184. IEEE.

Krippendorff, K. 2004. Content Analysis: An Introduction to Its Methodology Thousand Oaks. *Calif.: Sage* .

Kumar, A.; Sattigeri, P.; and Balakrishnan, A. 2017. Variational Inference of Disentangled Latent Concepts from Unlabeled Observations. *arXiv preprint arXiv:1711.00848* .

Lample, G.; Subramanian, S.; Smith, E.; Denoyer, L.; Ranzato, M.; and Boureau, Y.-L. 2019. Multiple-Attribute Text Rewriting. In *Proceedings of the International Conference on Learning Representations.*

Li, J.; Jia, R.; He, H.; and Liang, P. 2018. Delete, Retrieve, Generate: a Simple Approach to Sentiment and Style Transfer. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 1865–1874. Association for Computational Linguistics.

Logeswaran, L.; Lee, H.; and Bengio, S. 2018. Content Preserving Text Generation with Attribute Controls. In *Advances in Neural Information Processing Systems*, 5103–5113.

Louizos, C.; Swersky, K.; Li, Y.; Welling, M.; and Zemel, R. 2015. The Variational Fair Autoencoder. *arXiv preprint arXiv:1511.00830* .

Maaten, L. v. d.; and Hinton, G. 2008. Visualizing Data Using t-SNE. *Journal of Machine Learning Research* 9(Nov): 2579–2605.

Malandrakis, N.; Shen, M.; Goyal, A.; Gao, S.; Sethi, A.; and Metallinou, A. 2019. Controlled Text Generation for Data Augmentation in Intelligent Artificial Agents. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, 90–98. Association for Computational Linguistics.

Mathieu, E.; Rainforth, T.; Narayanaswamy, S.; and Teh, Y. W. 2018. Disentangling Disentanglement in Variational Autoencoders. *arXiv preprint arXiv:1812.02833* .

Moyer, D.; Gao, S.; Brekelmans, R.; Galstyan, A.; and Ver Steeg, G. 2018. Invariant Representations without Adversarial Training. In *Advances in Neural Information Processing Systems*, 9084–9093.

Narayanaswamy, S.; Paige, T. B.; Van de Meent, J.-W.; Desmaison, A.; Goodman, N.; Kohli, P.; Wood, F.; and Torr, P. 2017. Learning Disentangled Representations with Semi-supervised Deep Generative Models. In *Advances in Neural Information Processing Systems*, 5925–5935.

Preoţiuc-Pietro, D.; and Ungar, L. 2018. User-Level Race and Ethnicity Predictors from Twitter Text. In *Proceedings of the 27th International Conference on Computational Linguistics*, 1534–1545. Association for Computational Linguistics.

Pustejovsky, J.; Hanks, P.; Sauri, R.; See, A.; Gaizauskas, R.; Setzer, A.; Radev, D.; Sundheim, B.; Day, D.; Ferro, L.; et al. 2003. The Timebank Corpus. In *Corpus Linguistics*, volume 2003, 40. Lancaster, UK.

Rezende, D. J.; Mohamed, S.; and Wierstra, D. 2014. Stochastic Backpropagation and Variational Inference in Deep Latent Gaussian Models. *arXiv preprint arXiv:1401.4082* .

Rezende, D. J.; and Viola, F. 2018. Taming VAEs. *arXiv preprint arXiv:1810.00597* .

Romanov, A.; Rumshisky, A.; Rogers, A.; and Donahue, D. 2019. Adversarial Decomposition of Text Representation. 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)*, 815–825. Association for Computational Linguistics.

Sha, L.; Camburu, O.-M.; and Lukasiewicz, T. 2021. Learning from the Best: Rationalizing Predictions by Adversarial Information Calibration. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*.

Sha, L.; Shi, C.; Chen, Q.; Zhang, L.; and Wang, H. 2020. Estimating Minimum Operation Steps via Memory-based Recurrent Calculation Network. In *Proceedings of the 2020 international joint conference on neural networks (IJCNN). IEEE*.

Shao, H.; Yao, S.; Sun, D.; Zhang, A.; Liu, S.; Liu, D.; Wang, J.; and Abdelzaher, T. 2020. ControlVAE: Controllable variational autoencoder. In *Proceedings of the International Conference on Machine Learning*, 8655–8664. PMLR.

Shen, T.; Lei, T.; Barzilay, R.; and Jaakkola, T. 2017. Style Transfer from Non-Parallel Text by Cross-Alignment. In *Proceedings of the Advances in Neural Information Processing Systems*, 6833–6844.

Stühmer, J.; Turner, R. E.; and Nowozin, S. 2019. Independent Subspace Analysis for Unsupervised Learning of Disentangled Representations. *arXiv preprint arXiv:1909.05063* .

Xu, J.; Sun, X.; Zeng, Q.; Zhang, X.; Ren, X.; Wang, H.; and Li, W. 2018. Unpaired Sentiment-to-Sentiment Translation: A Cycled Reinforcement Learning Approach. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 979–988. Association for Computational Linguistics.

Zhao, J.; Kim, Y.; Zhang, K.; Rush, A. M.; and LeCun, Y. 2018. Adversarially Regularized Autoencoders. In *Proceedings of the 35th International Conference on Machine Learning*, 5897–5906. JMLR. org.