VAE based Text Style Transfer with Pivot Words Enhancement Learning

Haoran Xu [1,2], Sixing Lu [2], Zhongkai Sun [2], Chengyuan Ma [2], Chenlei Guo [2]
[1]Johns Hopkins University, [2]Amazon Alexa AI

hxu64@jhu.edu
{cynthilu, zhongkas, mchengyu, guochenl}@amazon.com

Abstract

Text Style Transfer (TST) aims to alter the underlying style of the source text to another specific style while keeping the same content. Due to the scarcity of high-quality parallel training data, unsupervised learning has become a trending direction for TST tasks. In this paper, we propose a novel VAE based Text Style Transfer with pivOt Words Enhancement leaRning (VT-STOWER) method which utilizes Variational AutoEncoder (VAE) and external style embeddings to learn semantics and style distribution jointly. Additionally, we introduce pivot words learning, which is applied to learn decisive words for a specific style and thereby further improve the overall performance of the style transfer. The proposed VT-STOWER can be scaled to different TST scenarios given very limited and non-parallel training data with a novel and flexible style strength control mechanism. Experiments demonstrate that the VT-STOWER outperforms the state-of-the-art on sentiment, formality, and code-switching TST tasks [1].

1 Introduction

Text style transfer (TST) is an important task in the natural language generation area, aiming to control the certain manner of the semantics style expressed in the generated text. Such styles include but not limit to emotion, humor, politeness, formality, and code-switching. For instance, sentiment transfer is widely seen in sentiment analysis for reviewing comments (e.g., yelp, twitter), and targets on converting the original negative/positive comment into a new comment with same topic but opposite sentiment (Hu et al., 2017; Shen et al., 2017); formality transfer is commonly used in documenting, aims at transferring the informal oral expression into a formal written expression (Jin et al., 2020). In this paper, we also consider code-switching as a style transfer task, which has not been explored by previous works. Code-switching is a complicated linguistic phenomenon where a speaker alternates between two or more languages in one utterance, either inter-sentential or intra-sentential. The code-switching transfer is a more challenging task considering cross-lingual alignment and limited available training data in nature. Examples of these three style transfer tasks are shown in the Figure 1.

Because of the scarcity of high-quality parallel training data, unsupervised learning has become the mainstream for TST tasks. Existing works on unsupervised TST learning can be roughly categorized into *Disentanglement* (Shen et al., 2017; Hu et al., 2017; Fu et al., 2018; John et al., 2019) and *Style Attribute Rewriting* (Lample et al., 2019; Dai et al., 2019; Yi et al., 2020). Disentanglement approaches strip style features from the content and incorporate the content features with the target style representation. However, researchers become less focus on disentanglement methods after Locatello et al. (2019) theoretically proved disentanglement approaches are impossible to represent style fully with unsupervised learning. The style attribute rewriting enforces the model to focus on style-independent words by cycle reconstruction and rewriting the style attributes with style embeddings. Dai et al. (2019) firstly proposed style transfer model based on the transformer architecture along with target style information. Lample et al. (2019) reported that a good decoder can generate the text with the desired style by rewriting the original style. However, the style strength of the generated sentences cannot be easily adjusted in above mentioned works.

Variational autoencoder (VAE) is firstly proposed by (Kingma and Welling, 2014) for gener-

Work done during an internship at Amazon Alexa AI.

[1]The code is available at https://github.com/fe1ixxu/VT-STOWER.

Proceedings of the 18th International Conference on Natural Language Processing, pages 162–172
Silchar, India. December 16 - 19, 2021. ©2021 NLP Association of India (NLPAI)

ation by formatting the latent distribution instead of feeding a single latent feature to the decoder. Many TST models have been benefited from the architecture of VAE. Bowman et al. (2016); John et al. (2019) showed that the latent space learned by VAE is considerably smoother and more continuous than the one learned by Deterministic Autoencoder (DAE). Hu et al. (2017) proposed a new neural generative model that combines VAE and holistic attribute discriminators for effective imposition of the style semantic structures. In this paper, we

Negative	they were **dry** and truly **tasteless**.
Positive	they were **tasty** and truly **delicious**.
Informal	**u 'll** find a man who really deserves **u** one day.
Formal	**you will** find a man who really deserves **you** one day.
Hindi	कया आपको एनीमशन **फिलम** पसद ह ? (Do you like animation movies?)
Code-switch	कया आपको एनीमशन **movies** पसद ह ? (Do you like animation movies?)

Figure 1: Examples of different TST. Including sentiment style transfer (negative ↔ positive), formality style transfer (informal ↔ formal), code-switch style transfer (single language ↔ code-switch sentence).

also leverage the VAE and propose a novel method called VAE based Text Style Transfer with pivOt Words Enhancement leaRn-ing (VT-STOWER) for TST tasks. VT-STOWER utilizes both VAE and style embeddings to jointly learn the distribution of content and style features. More importantly, we boost the performance of TST tasks much more by inventing pivot words enhancement learning. Compared with other style-transfer methods, our proposed VT-STOWER has a bunch of advantages. In general, the advantages and contributions of the VT-STOWER can be summarised as follows:

- VT-STOWER integrates the advantages of both VAE and style embeddings. The former catches continuous style expression distribution in language itself while the latter differentiates embedding between original style and target style.

- VT-STOWER has the flexibility to adjust the target style strength by granting different weights to the auxiliary target style embedding; This allows VT-STOWER to better migrate to different style transfer scenarios, which is rarely studied in previous style-transfer work.

- With the *pivot words masking enhancement* mechanism, VT-STOWER is able to focus more on the pivot words (certain words that can determine the style of the sentence) and be aware of which words have higher probability to be transferred in the TST. This enhancement significantly improves the transfer accuracy while maintaining original topic.

- VT-STOWER can be easily scaled to different types of TST tasks. To the best of our knowledge, we are the first to consider code-switching in perspective of style transfer and demonstrate that VT-STOWER can be successfully applied to the Hindi-Hinglish code-switching transfer. Therefore, we provide more potential solutions for the the code-switching problems beyond translation by which translating from single language to code-switching expression is very hard given limited training data.

- We evaluate VT-STOWER on the benchmark dataset of sentiment, formality transfer tasks, and the code-switching style transfer. Experimental results on all tasks demonstrate better overall performance against state-of-the-art methods, which highlights effectiveness and wide application of VT-STOWER.

2 Proposed Method

The training of VT-STOWER consists of two stages. The training stage I is a VAE reconstruction task in which the input text x will be reconstructed together with external style embeddings. The latent space of content distribution is learned by VAE, and the original and target style mapping will be learned and saved in style embeddings. The trained VAE and style embeddings will also be utilized in the second training stage.

To make the style transfer focus on pivot words (e.g., emotional words in sentiment TST) while maintaining other words unchanged (so that the fluency and semantics can be largely preserved), we fine-tune the VAE with pivot word masking in training stage II. The masking is based on the probability distribution of pivot words for specific styles, which is learned from a style classification task.

In the inference stage, VT-STOWER uses the learned external target style embeddings to adjust the sampled latent vector of the original input to the target style. The adjusted sentence vector will then be input to the decoder to generate the target style text.

2.1 Training Stage I: VAE & Style Embeddings

Figure 2a presents the details of training stage I. Given a sentence x whose style type is known, we firstly extract the contextualized vectors through a pre-trained language model as the input to the VAE model, since a pre-trained language model (such as `RoBerta` (Liu et al., 2019) and `XLM-R` (Conneau et al., 2020)) can improve the performance of the downstream models, especially when the training data size is small (Peters et al., 2018). After that, similar to typical VAE structure from Bowman et al. (2016); John et al. (2019), a multi-layer transformer is used as the encoder to encode x to a mean vector $u \in \mathbb{R}^d$ and a variance vector $\Sigma \in \mathbb{R}^d$ to construct a latent distribution $\mathcal{N}(\mu, \Sigma)$. d represents the dimension of the latent space. z is the vector sampled from the latent distribution and will be input to the decoder (which is also a multi-layer transformer) to reconstruct the original text. The latent distribution is assumed to be a normal distribution $\mathcal{N}(0, \mathrm{I})$. The standard loss function of the VAE model is defined as:

$$\mathcal{L}_{vae} = -\mathbb{E}_{q(z|x)}[\log p(x|z)] + \beta \cdot \mathbb{KL}(q(z|x) \parallel p(z)) \quad (1)$$

where the first term represents the likelihood of the reconstruction of the original text x while the second term is the Kullback–Leibler (KL) divergence between the latent distribution and standard normal distribution. $p(z)$ represents the prior which is the standard normal distribution $\mathcal{N}(0, \mathrm{I})$, and $q(z|x)$ is the posterior distribution in the form of $\mathcal{N}(\mu, \Sigma)$. β is the hyperparameter balancing the learning capacity between self-reconstruction and style features (Higgins et al., 2016).

Style embeddings are also learned in this training stage. Instead of disentangling style attributes from latent features (Shen et al., 2017; Hu et al., 2017; Fu et al., 2018; John et al., 2019), we utilize external style embeddings to learn the original and target style representations. The advantage of external style embeddings is that they can avoid separating latent feature which leads to the lower capacity of vector representation (Dai et al., 2019), and can differentiate the space of different styles. The set of style embeddings is defined as $S = \{s_1, s_2, \cdots, s_k\}, s_i \in \mathbb{R}^{k \times d}$, where k is the number of styles (k is commonly to be 2 in TST tasks). Style embeddings are generated by a linear forward network whose output dimension is d. This style embedding network is randomly initialized and will be updated by minimizing the similarity between the style embeddings and latent feature of the input instances.

To minimize such similarity, we calculate the cosine similarity between style embeddings s_i ($1 \leq i \leq k$) and sampled latent feature z as the style loss. The assumption is that the style embedding should be highly related to the latent feature encoded from the sentence which belongs to the same style, e.g., the distance between positive style embedding and latent vector encoded from positive sentence should be close to 1, while the distance should be 0 between positive style embedding and latent vector from the negative input sentence. The style loss is defined as follows:

$$\mathcal{L}_{style} = -\sum_{i=1}^{k} d_i \log(\sigma(\cos(s_i, \mathrm{sg}(z)))) \quad (2)$$

For brevity, we only present the loss for a single style sentence, where d_i represents the ground truth distance. Specifically, if i_{th} style is the style of the input sentence, $d_i = 1$, otherwise, $d_i = 0$. $\sigma(\cdot)$ here is the sigmoid activation function which controls the range of cosine similarity between 0 and 1. $\mathrm{sg}(\cdot)$ is the 'stop gradient' function, e.g., the feature $\mathrm{sg}(z)$ is extracted through the latent distribution and used as an independent constant vector for computing the $\mathcal{L}_{style}$. The VAE loss is slightly modified from Equation 1 by adding style embedding to hint decoder the style of sentences to be generated.

$$\mathcal{L}_{vae} = -\mathbb{E}_{q(z|x)}[\log p(x|z + \mathrm{sg}(s_x))] + \beta \cdot \mathbb{KL}(q(z|x) \parallel p(z)) \quad (3)$$

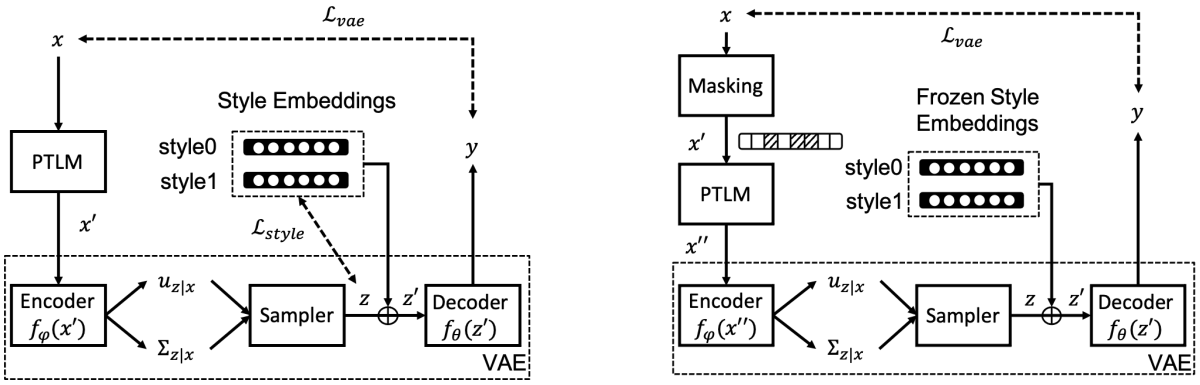
where s_x is the style embedding of sentence x. Similarly, the s_x is also used as a constant vector. Therefore, the total loss function is then defined as:

$$\mathcal{L}_{total} = \lambda_{vae}\mathcal{L}_{vae} + \lambda_{style}\mathcal{L}_{style} \quad (4)$$

where λ_{vae} and λ_{style} are penalty weights, which are hyperparameters to balance between VAE loss and style embeddings loss.

2.2 Training Stage II: Pivot Words Masking

In Stage I, we co-train VAE & style embeddings where we show how to leverage learned style embeddings to further improve the VAE model. In stage II, We further enhance the model by masking pivot words to prompt decoder to focus on pivot



(a) Training stage I: VAE & Style Embeddings

(b) Training stage II: Pivot Words Enhancement

Figure 2: Workflow of two training stages. a) Training stage I: VAE & style embeddings training. The VAE structure learns to reconstruct the inputs sentence x, and the style embeddings learn the vector representation of each style. PTLM represents Pre-Trained Language Model. b) Training stage II: pivot words masking training. The VAE is further fine-tuned with similar reconstruction task with additional pivot words masking. The frozen style embeddings are added to the latent vector to reconstruct the original sentence. We frozen the style embeddings in this step since the style-related pivot words have higher possibility to be masked and latent vector loses the style information which is the key for style embeddings training.

words, because certain style-related words play crucial roles in TST (Fu et al., 2019). For instance, the pivot word of the sentence 'I am disappointed with the restaurant' in sentiment transfer is 'disappointed' because this word contributes the most to the negative sentiment. However, other words such as "I, was" are anchor words, which are unrelated to the sentiment but affect the semantics thus should be unchanged during the style transfer. Therefore, this stage of training is important to enhance the model ability in transferring pivot words while keeping anchor words. This stage cannot be merged into training stage I because 1) in this stage style embeddings have no visibility to the style-related pivot words so that the style information is hard to be learned; 2) the style embeddings learned in training stage I have auxiliary function in helping reconstructing masked pivot words during fine-tuning the VAE.

However, randomly masking words in input sentence and only relying on style embeddings to emphasize the pivot words does not achieve ideal results. A more efficient way is to learn which words are more possible to be pivot words for a specific style, and mask them based on the probability. Similar to Sudhakar et al. (2019), we utilizes the importance score distribution to indicate the possibility of words being pivot (a pivot word has a higher score). Such importance score distribution is achieved from the attention weights of a style classifier. Specifically, we train a style classifier based on a pre-trained language model, appending with a softmax layer over the attention stack of the first token. The first token is usually a special symbol that represents the beginning of the sentence (e.g., '<s>'), and also collects other tokens' attention weights that correspond to their significance in identifying the style of the input sentence. The importance score of a token w in the input sentence x is defined as follows:

$$\alpha(w) = \frac{1}{L}\sum_{i=1}^{L} \text{softmax}_{w\in x}\left(\frac{Q_{<s>,i}K^T_{w,i}}{\gamma}\right) \quad (5)$$

where L is the number of attention heads. Q, K are quires and keys in the final layer of the language model (Vaswani et al., 2017). Their subscript $< w, i >$ represents the vector of token w in i_{th} head. γ is a hyperparameter ranging in (0,1) to adjust the sharpness of the score distribution (smaller means sharper).

After we get the pivot words probability, we mask words in the input sentences based on this importance score distribution. Specifically, every token x_i is assigned a random number p_i, conforming to the uniform distribution $p_i \sim$ uniform[0,1]. Tokens are masked into a special symbol '<mask>' if their assigned number is smaller than the score ($p_i < \alpha(x_i)$) so that words that possess higher important scores have higher probability to be masked. Following the previous example, the input sentence would be masked as 'I was <mask> with the restaurant'. In this way, masked sentence pre-

serves the content but with style attributes removed. Then the VAE model is fine-tuned to reconstruct masked sentence to the original sentence by adding the corresponding style embedding to the latent feature. The loss function is defined as follows:

$$\mathcal{L}_{vae} = -\mathbb{E}_{q(z|x)}[\log p(x|z + \text{sg}(s'_x))] + \beta \cdot \mathbb{KL}(q(z|x) \parallel p(z)) \quad (6)$$

where s'_x is the style embedding which has the same style as input x. Note that we do not update style embeddings in this stage because the style embeddings are used for assisting fine-tuning the decoder with style information, and their general style representation should not be impacted by the pivot words reconstruction loss. Moreover, to prevent the latent space of VAE from shifting or distorting to unreasonable distribution that only describes masked sentences, we conduct pivot words masking in randomly 50% of sentences.

Although Madaan et al. (2020) has a similar method tagging the source style phrases and generating the target style sentences by using n-gram tf–idfs, the core differences of our stage II method are: 1) each word has a probability of being masked calculated by the attention scores of the stacked classifier on a language model, which leads to a smooth word masking probability distribution; 2) VAE decoder reconstructs the masked sentences using both information of latent space and external style embeddings.

2.3 Inference stage

In the inference stage, the latent representation z generated from the input sentence x through VAE will be adjusted before sending to the decoder. In detail, the latent vector z will be added the target style embedding and subtracted the style embedding of original style as x. Intuitively, we expect the injection and removal of style information is completed by the addition and subtraction operations of style embeddings. The updated latent representation is expressed as follows:

$$z' = z + w \cdot (s_t - s_o) \quad (7)$$

where s_t and s_o are target and original style embeddings trained in the stage I respectively. w represents the style weight that adjusts the style strength applied to the sentence generation. A higher weight means stronger style attributes will be injected for generation.

3 Experiments

3.1 TST Evaluation Tasks and the Settings

We evaluate VT-STOWER with three different TST tasks: sentiment transfer, formality transfer, and code-switching transfer.

For sentiment transfer, we adopt the Yelp dataset (Li et al., 2018), in which each sample is a business review of a restaurant and is labeled as positive or negative. For formality transfer, We adopt one of the largest corpus for formality transfer task, namely *Family and Relationships* domain data in GYAFC (Grammarly's Yahoo Answers Formality Corpus) (Rao and Tetreault, 2018). For code-switching transfer, we evaluate VT-STOWER on a Hindi→Hinglish transfer task, which is extracted from the English-Hinglish translation dataset at LinCE (Linguistic Code-switching Evaluation) (Aguilar et al., 2020). We first translate English sentences into Hindi by Amazon Translation Service and then transliterate Latin scripts of Hindi words into Devanagari form by using *indic-trans* tool (Bhat et al., 2014) to keep the consistency of the script of language [2]. Note this dataset is very low-resource, which only contains 7K sentences for training. Similar to GYAFC set, we shuffle the training data and treat it as unpaired data. Note that two of the training sets are transformed from originally paired dataset instead of directly using richer unpaired datasets, it is because we want to make a fair comparison with other referenced approaches. The training and test set size for each task is presented in Table 1.

Tasks	training set	evaluation set	test set
Sentiment Transfer (positive/negative)	266K/177K	2K/2K	500/500
Formality Transfer (formal/informal)	52K/52K	2.2K/2.7K	1K/1.3K
Code-Switching Transfer (Hinglish/English)	7K/7K	300/300	300/300

Table 1: Training, evaluation, and test set size of three evaluation tasks.

Considering that GYAFC and Yelp dataset are written in English and the code-switching dataset is in mixed languages of Hindi and English, we use `RoBERTa` as the pre-trained language model for sentiment and formality transfer tasks, and `XLM-R` (Conneau et al., 2020) for code-switching transfer. Also, we fine-tune the style classifier to obtain important score distribution by leveraging `RoBERTa` and `XLM-R` for the corresponding transfer tasks.

[2]The output of translator is Devanagari form while the original script of Hindi in LinCE is Latin.

More training hyperparameters are shown in Appendix A.

3.2 Evaluation Metrics

Style Transfer Accuracy (Acc) Style transfer accuracy (Acc) is defined as the ratio of the number of successfully transferred sentences and the total number of input sentences. Following previous studies (Dai et al., 2019; Sudhakar et al., 2019), we leverage fastText classifier (Joulin et al., 2017) to classify whether the original text have been successfully transferred to the target style. The classifier is trained on the same training data used for style transfer. The three classifiers achieve 97.6%, 85.75% and 99.7% accuracy for sentiment, formality and code-switching style classification itself, respectively.

Perplexity (PPL) We also measure the fluency of the transferred sentences by calculating their perplexity. The lower the perplexity is, the more fluent the generated sentences are. For the GYAFC and Yelp dataset which are in English, we use the pre-trained language model GPT2 (Radford et al., 2019) to compute the perplexity, where no further fine-tuning is conducted. However, GPT2 does not apply to other languages or code-switching sentences. Following Samanta et al. (2019), we train a character-level LSTM (Kim et al., 2016) on the code-switching training data and utilize this model to derive the perplexity of generated code-switching sentences.

BLEU Scores Content preservation is evaluated by the tokenized BLEU scores (Papineni et al., 2002) between the transferred sentences and human-authored references, which is calculated with the `multi-bleu.perl`. Note that GYAFC dataset has four human references, so the BLEU for GYAFC is the mean BLEU scores between the generated sentences and four references. Because there is no human reference for code-switching task, we report BLEU scores between transferred sentences and original sentences for code-switching transfer instead.

Geometric Mean (GM) Following Yi et al. (2020), We also report the geometric mean of accuracy, BLEU, $\frac{1}{\ln \text{PPL}}$ as the overall performance.

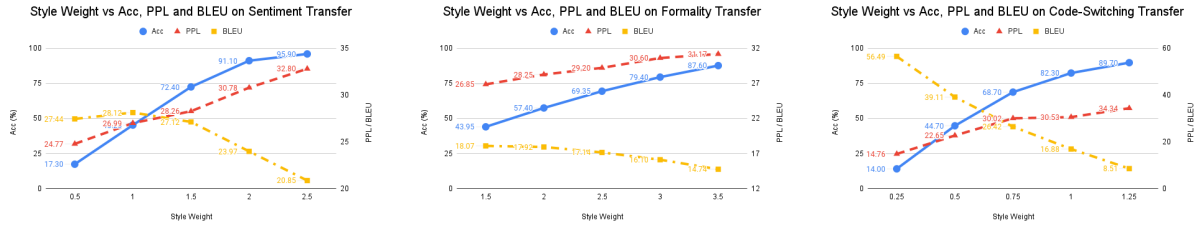
3.3 Main Results

The performance of VT-STOWER and previous works are shown in Table 2. First of all, we can clearly see the performance improvement brought by stage II training compared with single training stage I. **In all three transfer tasks, models trained in stage II lead to lower PPL (or similar PPL in code-switching transfer) and higher BLEU scores when we find a w to control them in a similar Acc, which achieves better overall performance.** For instance, compared with the model trained in stage I with $w = 4$ in the sentiment transfer, the model fine-tuned in stage II achieves similar accuracy with $w = 2$ (91.7% vs. 91.1%). At the same time, the stage II model decreases the PPL from 38.35 to 30.78 and increases the BLEU from 18.51 to 23.97, which demonstrates that the pivot words masking training is capable of improving the smoothness of the sentences and the preserving the content. Note that we cannot use the same weight w for a direct comparison since the models in two training stages have different sensitivity to w. Therefore, we use w that produces similar accuracy between stage I and stage II for a fair comparison for the PPL and BLUE.

When comparing our method with several state-of-the-art references: CrossAlignment (Shen et al., 2017), Delete & Retrieve (Li et al., 2018), B-GST (Sudhakar et al., 2019), Style Transformer (Dai et al., 2019), Deep LatentSeq (He et al., 2020), Tag & Generate (Madaan et al., 2020) and StyIns

Models	Acc ↑	PPL ↓	BLEU ↑	GM ↑
Sentiment Transfer (Yelp)				
CrossAlignment	74.0	42.91	9.06	5.63
Delete & Retrieve	87.5	40.66	5.99	5.21
B-GST	84.3	**25.27**	22.82	8.41
Style Transformer	83.9	43.60	**28.29**	8.57
Deep LatentSeq	83.0	27.08	24.03	8.46
StyIns	91.5	42.60	25.11	8.49
Tag & Generate	87.5	32.98	21.80	8.17
Ours (stage I, $w = 4$)	**91.7**	38.35	18.51	7.75
Ours (stage II, $w = 2$)	91.1	30.78	23.97	**8.61**
Human Reference	74.1	27.40	100.0	13.08
Formality Transfer (GYAFC)				
CrossAlignment	65.35	**13.66**	1.57	3.40
Delete & Retrieve	53.85	29.70	11.71	5.71
Style Transformer	56.05	48.72	**24.67**	7.09
Ours (stage I, $w = 4$)	80.9	31.90	14.19	6.92
Ours (stage II, $w = 3.1$)	**81.0**	30.78	15.84	**7.21**
Human Reference	82.31	28.05	100.0	13.39
Code-Switching Transfer (LinCE)				
Style Transformer	**99.3**	601.45	3.47	3.78
Randomly Replace	1.02	213.24	**69.09**	2.36
Ours (stage I, $w = 0.75$)	66.67	29.91	24.30	7.81
Ours (stage II, $w = 0.75$)	68.70	30.02	26.42	**8.11**

Table 2: Overall results of our models (VT-STOWER) and previous methods on three style transfer tasks. The best scores are bolded in the corresponding metric. ↑ means the higher is better, vice versa.



(a) sentiment transfer (b) formality transfer (c) code-switching transfer

Figure 3: Illustration of style weight w vs. Acc, PPL and BLEU in sentiment, formality and code-switching transfer tasks. Note there is a trade-off between Acc and PPL/BLEU. With increasing of w, Acc will increase while BLEU drops down and PPL increases.

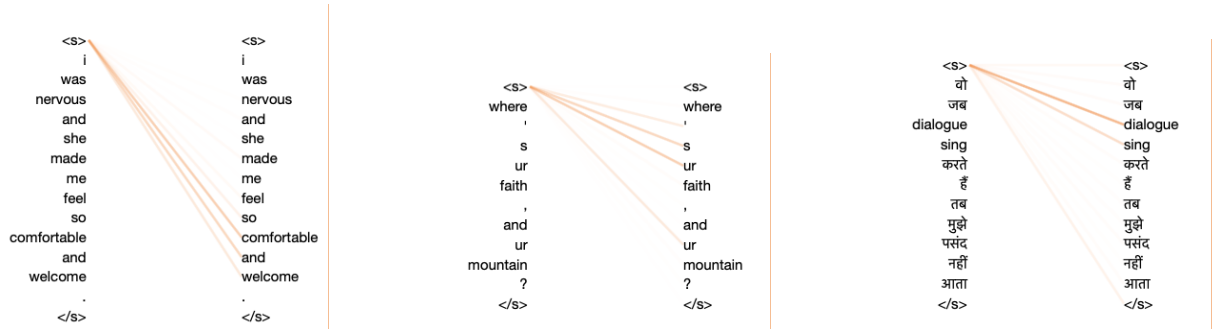
Negative → Positive Transfer (Yelp)	
Original	the place has obviously gone downhill over the years .
Style Transformer	the place has consistently gone handful over the years .
VT-STOWER	the place has greatly improved over the years .
Informal → Formal (GYAFC)	
Original	ask her out or tell her u like or admire her .
Style Transformer	ask her out or tell her is like or admire her .
VT-STOWER	ask her out or inform her that you like her .
Code-Switching Transfer (LinCE)	
Original	म सहमत ह ! डालता ह । मझ यकीन नही था कि अत म कस महसस किया जाए । (I agree! I was not sure how to feel so.)
Style Transformer	movie की हैं (are movie)
Randomly Replace	म सहमत ह ! डालता ह । मझ यकीन no had कि अत म tight महसस किया जाए । (I agree! I put I sure had no idea that I could be felt so tight.)
VT-STOWER	i agree ! मैं sure नहीं था की end मैं कैसा feel करू (i agree ! I wasn't sure how would I feel)

Figure 4: Comparison of style transfer outputs of our models and style transformer in three transfer tasks. Our models are stage II models in Table 2. Translations for code-switching sentences are shown in parenthesis.

(Yi et al., 2020), the performance of their methods is directly evaluated on their provided outputs by using our metric evaluators. We will further discuss how w affect the performance in next section. We can clearly see the overall performance (GM) of our proposed model is better than all baselines. For evaluating the success of style transfer, accuracy is the most critical metrics, for which VT-STOWER also demonstrates large improvement in sentiment and formality transfer.

In the sentiment style transfer, our model with $w = 2$ (after stage II training) has competitive accuracy (91.1%), and BLEU (23.97) compared with the state-of-the-art methods StyIns (Yi et al., 2020) (accuracy=91.5%, BLEU=25.41) and style transformer (Dai et al., 2019) (accuracy=83.9%, BLEU=28.29) but achieve much lower perplexity (30.78) compared to 42.60 in StyIns and 43.60 in style transformer, which demonstrates that the sentences generated from our model is closer to the natural language. VT-STOWER also outperforms other previous methods by a large margin in all three metrics. In the formality transfer, the most competitive model is the style transformer. Although it achieves higher BLEU scores (24.67), our models beats it on higher style transfer accuracy (81.0% vs. 56.05%) and significantly lower PPL (30.78 vs. 48.72) with limited loss of BLEU scores.

For the code-switching transfer, since there is no previous TST experimenting on this task, we train the strongest baseline (style transformer) for this task. Interestingly, style transformer obtains a very high accuracy (99.8%) with the costs of very high PPL (601.46) and very low BLEU score (3.47). The possible reason is that the style transformer is only able to capture partial special style features from the small dataset (7K) and only transfer sentences based on these features without fully capturing the nature of languages, resulting in high accuracy but low fluency and BLEU. However, VT-STOWER can balance among the accuracy, fluency, and BLEU to achieve reasonable results even in the case of the low-resource dataset, which demonstrates its generalization power. Additionally, we also design another baseline, i.e., we randomly replace 15% Hindi words with English words (Zheng et al., 2021) based on the MUSE dictionary (Conneau et al., 2017), because intuitively, people may regard code-switching text generation as simply translating several words. However, this method only achieve 1.02% accuracy, because simple translation and replacement cannot accord with the habit of bilingual expression in code-switching sentences, namely, code-switching has its own style according to the speakers (e.g., usually noun is more likely to be replaced with foreign language than preposition). Intuitively, when we compare the VT-STOWER with original and other approaches as shown in Figure 4, the output bilingual sentence from VT-STOWER reads more fluent and can be easier understood.



(a) sentiment transfer (b) formality transfer (c) code-switching transfer

Figure 5: Illustration of the mean attention weights of token '<s>' from all heads at the final layer in three TST tasks. Higher importance scores are assigned to pivot words, which are depicted as deeper lines in the figures.

3.4 Effect of Style Weights

As shown in Equation 7, the strength of the target style in z' is adjusted by the style weight w. In Figure 3, we present metrics trend with five different w for the models trained in stage II [3], and demonstrate how the style weight w affects the outputs. Taking sentiment transfer task as an example, when w is increased from 0.5 to 2.5, the transfer accuracy climbs from 17.3% up to 95.9%, but the BLEU score drops from 27.44 down to 20.85, and PPL increases from 24.77 to 32.8. The reason is when increasing w, more style information is injected into the latent vector that the decoder pays more attention to the target style feature rather than the naturalness and content of generated sentences. Therefore, w is a trade-off hyperparameter between the transfer accuracy and PPL/BLEU. Examples of generated sentences transferred from positive to negative sentiment with $w = 1.5, 2, 2.5$ are illustrated in Table 3. When $w = 1.5$, the model still can find a positive word, 'enjoying,' which makes the sentence ironical. In the case of $w = 2$, the 'enjoying' is rephrased to 'avoid', turning the sentence into a full negative attitude. If we further increment $w = 2.5$, more negative words will be added regardless of the smoothness of the sentence. Similar discussions also hold for the formality and code-switching transfer, where their results versus various style weights are illustrated in Figure 3b and 3c.

3.5 Importance Score Distribution

Recall that for the training stage II, the importance scores are derived from the attention scores of the BOS token '<s>', which are the mean scores of all heads from the last layer of a pre-trained encoder. Figure 5 presents the examples of importance score, showing how the score value represents the importance of words in terms of style representation. The importance scores are higher on 'comfortable and welcome' in the sentiment transfer, these words represents strong positive emotions. Similarly, the scores are higher on the informal written words 'ur' in the formality transfer, and English words mixed in a Hinglish sentence in the code-switching transfer.

[3] w ranges from 0.5 to 2.5 with an interval of 0.5 for sentiment and formality transfer, and from 0.25 to 1.25 with an interval of 0.25 for code-switching transfer.

Positive → Negative with various w	
Original	i will be going back and enjoying this great place !
$w = 1.5$	i will be going back and enjoying this **terrible** place !
$w = 2$	i will be going back and **avoid** this **terrible** place !
$w = 2.5$	i will be going back and **worst rude avoid** this **terrible** place !

Table 3: Examples of sentences transferred from positive to negative sentiment with various settings of w. The higher w is the more negative words are injected in the sentences.

4 Conclusion

We proposed the VT-STOWER, a model joinly trained with VAE and style embeddings for content distribution and style information. The method successfully transfers several different text styles, including the code-switching TST task for the first time. Taking advantage of the flexibility of style embeddings, our proposed model has the ability to adjust the style strength during the transfer by simply adjusting the style weights. To further enhance the transfer accuracy, we propose additional pivot words masking training scheme, which shows impressive improvement.

References

Gustavo Aguilar, Sudipta Kar, and Thamar Solorio. 2020. LinCE: A Centralized Benchmark for Linguistic Code-switching Evaluation. In Proceedings of The 12th Language Resources and Evaluation Conference, pages 1803–1813, Marseille, France. European Language Resources Association.

Irshad Ahmad Bhat, Vandan Mujadia, Aniruddha Tammewar, Riyaz Ahmad Bhat, and Manish Shrivastava. 2014. Iiit-h system submission for fire2014 shared task on transliterated search. In Proceedings of the Forum for Information Retrieval Evaluation, FIRE '14, page 48–53, New York, NY, USA. Association for Computing Machinery.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning, pages 10–21, Berlin, Germany. Association for Computational Linguistics.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Édouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 8440–8451.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. arXiv preprint arXiv:1710.04087.

Ning Dai, Jianze Liang, Xipeng Qiu, and Xuanjing Huang. 2019. Style transformer: Unpaired text style transfer without disentangled latent representation. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 5997–6007, Florence, Italy. Association for Computational Linguistics.

Yao Fu, Hao Zhou, Jiaze Chen, and Lei Li. 2019. Rethinking text attribute transfer: A lexical analysis. In Proceedings of the 12th International Conference on Natural Language Generation, pages 24–33.

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. 2018. Style transfer in text: Exploration and evaluation. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 32.

Junxian He, Xinyi Wang, Graham Neubig, and Taylor Berg-Kirkpatrick. 2020. A probabilistic formulation of unsupervised text style transfer. In International Conference on Learning Representations.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2016. beta-vae: Learning basic visual concepts with a constrained variational framework.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In International Conference on Machine Learning, pages 1587–1596. PMLR.

Di Jin, Zhijing Jin, Zhiting Hu, Olga Vechtomova, and Rada Mihalcea. 2020. Deep learning for text style transfer: A survey. arXiv preprint arXiv:2011.00416.

Vineet John, Lili Mou, Hareesh Bahuleyan, and Olga Vechtomova. 2019. Disentangled representation learning for non-parallel text style transfer. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 424–434, Florence, Italy. Association for Computational Linguistics.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, pages 427–431, Valencia, Spain. Association for Computational Linguistics.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In Thirtieth AAAI conference on artificial intelligence.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

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

Guillaume Lample, Sandeep Subramanian, Eric Smith, Ludovic Denoyer, Marc'Aurelio Ranzato, and Y-Lan Boureau. 2019. Multiple-attribute text rewriting. In International Conference on Learning Representations.

Juncen Li, Robin Jia, He He, and Percy Liang. 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), pages 1865–1874, New Orleans, Louisiana. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692.

Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Rätsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. 2019. Challenging common assumptions in the unsupervised learning of disentangled representations. In ICML.

Aman Madaan, Amrith Setlur, Tanmay Parekh, Barnabas Poczos, Graham Neubig, Yiming Yang, Ruslan Salakhutdinov, Alan W Black, and Shrimai Prabhumoye. 2020. Politeness transfer: A tag and generate approach. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 1869–1881, Online. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In Proceedings of the 40th annual meeting of the Association for Computational Linguistics, pages 311–318.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. OpenAI blog, 1(8):9.

Sudha Rao and Joel Tetreault. 2018. Dear sir or madam, may I introduce the GYAFC dataset: Corpus, benchmarks and metrics for formality 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), pages 129–140, New Orleans, Louisiana. Association for Computational Linguistics.

Bidisha Samanta, Sharmila Reddy, Hussain Jagirdar, Niloy Ganguly, and Soumen Chakrabarti. 2019. A deep generative model for code-switched text. IJCAI.

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

Akhilesh Sudhakar, Bhargav Upadhyay, and Arjun Maheswaran. 2019. "transforming" delete, retrieve, generate approach for controlled text style transfer. 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), pages 3269–3279.

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 Advances in neural information processing systems, pages 5998–6008.

Xiaoyuan Yi, Zhenghao Liu, Wenhao Li, and Maosong Sun. 2020. Text style transfer via learning style instance supported latent space. In Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20, pages 3801–3807. International Joint Conferences on Artificial Intelligence Organization. Main track.

Bo Zheng, Li Dong, Shaohan Huang, Wenhui Wang, Zewen Chi, Saksham Singhal, Wanxiang Che, Ting Liu, Xia Song, and Furu Wei. 2021. Consistency regularization for cross-lingual fine-tuning. arXiv preprint arXiv:2106.08226.

A Hyperparamerters

The encoder and decoder for sentiment and formality transfer tasks both are two-layer transformer (Vaswani et al., 2017), with FFN dimension size of 1024 and 4 attention heads. Due to the limited code-switching data size, we run smaller encoder and decoder with 256 FFN dimensions and 2 attention heads for code-switching transfer task. The neural networks that formulate the mean and variance of the latent space are also one-layer transformer blocks. The dimension of the latent features and style embeddings is 768. Both penalty weights λ_{vae} and λ_{style} equal to 1. We set γ as 0.01, 0.03, and 0.005 for sentiment, formality, code-switching transfer during importance score calculation, respectively. The β is set as 1. The optimizer is Adam (Kingma and Ba, 2014) with learning rate 0.0005. The batch size is 8092 tokens.