How Positive Are You: Text Style Transfer using Adaptive Style Embedding

Heejin Kim[1] **and Kyung-Ah Sohn**[1,2*]
[1]Department of Artificial Intelligence
[2]Department of Software and Computer Engineering
Ajou University
{heeeee123, kasohn}@ajou.ac.kr
* Corresponding author

Abstract

The prevalent approach for unsupervised text style transfer is disentanglement between content and style. However, it is difficult to completely separate style information from the content. Other approaches allow the latent text representation to contain style and the target style to affect the generated output more than the latent representation does. In both approaches, however, it is impossible to adjust the strength of the style in the generated output. Moreover, those previous approaches typically perform both the sentence reconstruction and style control tasks in a single model, which complicates the overall architecture. In this paper, we address these issues by separating the model into a sentence reconstruction module and a style module. We use the Transformer-based autoencoder model for sentence reconstruction and the adaptive style embedding is learned directly in the style module. Because of this separation, each module can better focus on its own task. Moreover, we can vary the style strength of the generated sentence by changing the style of the embedding expression. Therefore, our approach not only controls the strength of the style, but also simplifies the model architecture. Experimental results show that our approach achieves better style transfer performance and content preservation than previous approaches.[1]

1 Introduction

Text style transfer is the task of modifying a text with a specified style attribute. Given a source text with an attribute (e.g., positive), the text style transfer problem aims to alter this input text to achieve a different attribute (e.g., negative). The significant concern in this problem is that the generated text must meet some requirements: the text must (i) reflect the given target attribute, (ii) maintain the content that is style-irrelevant, and (iii) generate a sentence that appears natural.

Text style transfer has been solved by a supervised method using a parallel dataset (Jhamtani et al., 2017) containing pairs of source and target sentences. However, obtaining this parallel dataset that achieves a total one-to-one correspondence with a specified style is often not possible. Therefore, unsupervised approaches have recently been actively researched. The first among these unsupervised approaches is disentanglement. To satisfy the first two style transfer requirements, namely, reflecting the style attributes and preserving the contents, the disentanglement approach tries to perfectly separate the style component from the content component within the input text. If the input is completely separated, the text style is transferred by substituting the style component with a new target; for this approach, a discriminator is used to separate the style from a latent representation z, a compressed representation of the input text (Fu et al., 2018; John et al., 2019). The discriminator can also be applied to the output text so that the output text reflects the given attribute (Shen et al., 2017; Hu et al., 2017; Zhao et al., 2018). Nevertheless, this approach has weaknesses. First, dividing a sentence representation into style and content is difficult because these two components are not mutually exclusive. Furthermore, this separation induces a loss of information.

[1]https://github.com/kinggodhj/How-Positive-Are-You-Text-Style-Transfer-using-Adaptive-Style-Embedding

Proceedings of the 28th International Conference on Computational Linguistics, pages 2115–2125
Barcelona, Spain (Online), December 8-13, 2020

Another approach is to rewrite the style attribute of the input text. This approach does not perfectly disentangle the style from the text representation; rather, it uses extra style information along with a text representation. If style information is added to the generation model, the output sentence reflects the inserted style attribute (Lample et al., 2019). Moreover, a style classifier, which determines the style of the sentence, is applied to an output sentence to enhance the transfer performance (Dai et al., 2019). However, these approaches assign two jobs to the generation model: the first is to reconstruct the input sentence, and the second is to transfer the style. When the sentence generation model is also in charge of style expression, the style expression cannot be properly learned.

Moreover, these previous approaches cannot address the style strength because the meaning of the style is not learned. Therefore, ideally, we divide the architecture of the style transfer model into two modules, namely, sentence reconstruction and style representation modules. This means that the reconstruction model is no longer responsible for converting the style. As a result, the generation model can concentrate on generating sentences, while the style model can focus on learning direct style expressions. Therefore, the proposed model directly learns the representations of style expressions.

Figure 1 presents the main differences between previous models and the proposed model. Our model consists of two modules, namely, the transformer-based autoencoder and the style embedding module. In our architecture, style embedding is learned in the style embedding part, not in the autoencoder part; hence, the autoencoder part can concentrate on restoring the input. Therefore, our model is more concise than previous models and uses only one reconstruction loss to train the autoencoder model. To learn the style representation, the similarities between the style embedding and the latent text representations are used as input to the style classifier. Because the similarity is calculated through dot products with input latent representation, the learned style embedding also follows the expression of the input representation in the latent space. As a result, the style embedding that expresses the general representation of each style is learned. This style embedding module does not affect the autoencoder module, and the style embedding part can also concentrate on learning the style representation.

In addition, the separation of the model into a style part and a reconstruction part makes it possible to control the strength of the style in the output text. The style embedding part learns the general representation of each style. With this style embedding, the degree of style modification can be adjusted. We can also generate the output with an expression stronger than the input through the same style as the input. Because our model can control how much of the style is changed, it is characterized by its multimodality

In summary, our main contributions are as follows. (i) We adjust the style strength of the output, which was not previously possible; therefore, the output sentences can be multimodal. (ii) The autoencoder model for latent text representation is more concise and has fewer constraints than the models in previous studies. Each model can concentrate on its task (sentence generation and style representation). Ultimately, our model achieves competitive performance for style accuracy and sentence retention compared to other models.

2 Related Work

Label Embedding. Label embedding is used in many text-based problems. In machine translation, a token of the target language is used at the beginning of the input sentence to provide information regarding the target language (Johnson et al., 2017). The generation model learns this token representation automatically and provides a clue as to which language to translate at the test time. Using this information, Johnson et al. (2017) built a single model to translate multiple languages. Label embedding can also be used in text classification problems. Each word is represented by a word embedding vector, and the label of the sentence (such as positive or negative) is trained based on the similarities with word vectors. Label information is added as a weight to represent the word vectors, which are newly modified with this label information (Wang et al., 2018). This modification of word vectors using label embedding improves the text classification performance. In domain adaption, Li et al. (2019) used domain-specific style labels to give information on certain domains; with these domain labels, they can transfer the style among multiple datasets. Additionally, in text style transfer, style label embedding is used to provide information about transferring attributes for the generation model (Dai et al., 2019).

Text style transfer. Like many other cross-domain problems, the main problem of style transfer is that parallel data are limited. The first approach to handle this issue is disentanglement. This approach uses a discriminator to extract style attributes from the text information. When the discriminator is applied to the output text and the attribute is given, the discriminator makes the output involve the given attribute. This means aligning the output text to the attribute domain (Shen et al., 2017; Hu et al., 2017; Zhao et al., 2018). Alternatively, the discriminator is applied after the latent representation. At this location, the discriminator removes the attribute information from the latent space. At the test time, the style attribute is newly inserted into the decoder, and thus, the generated output text contains the given target style attribute (Fu et al., 2018; John et al., 2019). Additionally, the discriminator is pretrained to detect style words in the text, and these detected attribute words are eliminated from the sentence. At the inference phase, the other domain's style words are added into the non-style text (Li et al., 2018; Xu et al., 2018; Sudhakar et al., 2019).

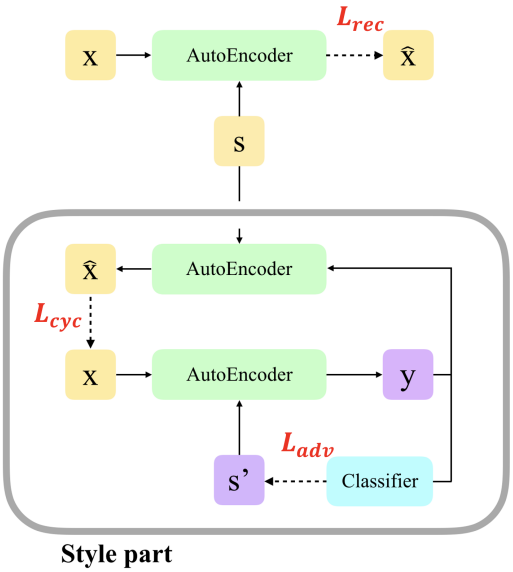
Unfortunately, fully separating the style and content is difficult and often requires the pretraining phase. Therefore, another approach that does not employ disentanglement has been proposed. This technique does not extract style information from latent representations but rather overwrites the transferred sentence using the style attribute corresponding to the user's intent (Lample et al., 2019; Dai et al., 2019). To reflect the style, this approach uses extra constraints, such as back-translation loss and adversarial style loss (e.g., the disentanglement approach). The back-translation loss (Sennrich et al., 2016) is similar to the cycle loss in the image domain (Zhu et al., 2017), which enhances the rewriting performance through the preservation of content. For instance, the input sentences are in two domains (X and Y), and each domain has a corresponding style s, s'. The input text x in the X domain is transferred to $\hat{y}$ by $D(E(x), s')$=$\hat{y}$ where E and D represent an encoder and a decoder, respectively. The style is modified once again with the transferred output $\hat{y}$ by $D(E(y), s)$=$\hat{x}$. The difference between the original input x and the final output $\hat{x}$ is minimized, which prevents the loss of content in the text. However, considerable time is required for training because the transfer procedure must be performed twice. Second, the style adversarial loss provides an indirect guide to the generation model to transfer the input sentence. This guidance constrains the generation model to follow the style information in the label embedding rather than the latent text representation. However, label embedding does not represent style expressions directly. Therefore, models with this architecture are limited to generating output sentences with only a single modality. Even if the same sentence is converted multiple times, the exact same result is obtained each time without any variation. Additionally, similar to the disentanglement approach, this adversarial loss hinders the generation model from focusing only on the sentence reconstruction. Thus, we propose a method that does not require a discriminator or model constraints, such as the back-translation loss, pretrained classifiers or pretrained autoencoders, but instead employs rewriting with style label embedding.

3 Method

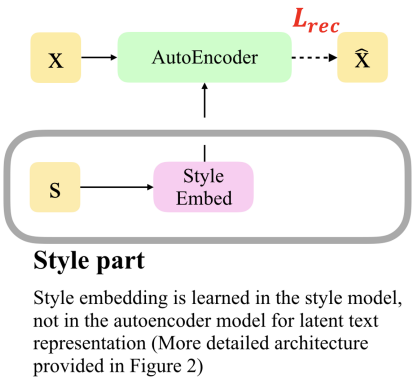
The architecture of our proposed model is presented in Figure 2. The model is divided into two parts: the autoencoder model for the content, and the style embedding model. In the autoencoder model, the latent representation from the encoder and the style representation are combined, and the decoder uses the resulting expression to generate the output text. During training, the autoencoder model's objective is to reconstruct the input text using the combined latent representation. The style embedding model is responsible for controlling the style of the input text. The style embedding model learns the representation for each style, and the classifier predicts the style using this embedding information. The objective of this part is to learn the style expression classifying the style label. The separation of these two parts simplifies the generation model and makes it possible to use adaptive style expressions. Additionally, each model can concentrate solely on its task.

3.1 Model architecture

Transformer based Autoencoder Most text generation tasks are based on a sequence-to-sequence model, which consists of an encoder and a decoder (Johnson et al., 2017; Lample et al., 2018). We



(a) Conventional overwriting-based architectures in previous approaches



(b) Our model with style embedding

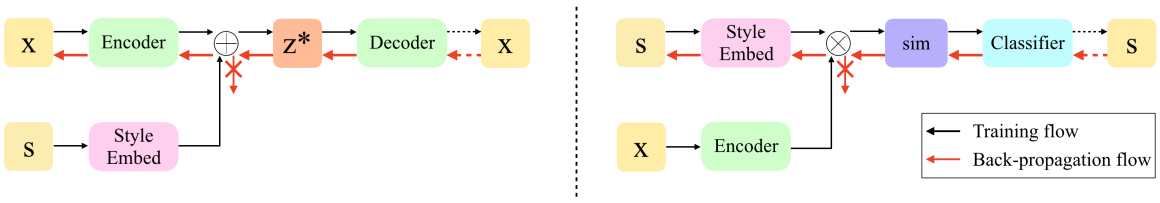
Figure 1: Comparison between models at the training phase. **(a)** Most style overwriting-based models are trained by the combination of three losses: reconstruction, cycle, and adversarial losses. As a result, many tasks are assigned to a single generation model. Additionally, the task is to make the output y follow the given target style s', not to learn the style expression. Therefore, most overwriting models cannot control how much style is changed in the text. **(b)** Our model is trained by the reconstruction loss only, and this simplicity can make the model focus more on generating sentences. Learning style expressions and adding them to the latent text representation makes it possible to control the output attribute.

use the Transformer-based sequence-to-sequence model to capture the various meanings of the same word. That is, the same word can have slightly different meanings in sentimental text; to capture these subtle differences, relation information is used in Transformer (Vaswani et al., 2017). In our approach, the transformer-based autoencoder model is trained to reconstruct the input text x with its own style s. Pairs of x, s' and x', s are not used in training phase. The generation model does not need to make the output follow a given attribute like other methods (Shen et al., 2017; Hu et al., 2017; Dai et al., 2019). Therefore, we do not need to add constraints such as the adversarial loss or cycle loss. Instead, the generation model focuses only on the reconstruction when the combined latent representation is given. To train this plain autoencoder, we use label smoothing regularization to improve the performance (Szegedy et al., 2016). The loss of the autoencoder model is expressed as follows:

$$L_{rec}(D_{\theta_d}(E_{\theta_e}(x), x) = -(1-\epsilon)\sum_{i=1}^{v}\bar{p}_i \log(p_i) + \epsilon/v \sum_{i=1}^{v}\log(p_i), \quad (1)$$

where v is the vocabulary size and ϵ represents the smoothing parameter. p and $\bar{p}$ denote the predicted probability distribution and the true probability distribution over the vocabulary, respectively. This reconstruction loss does not affect the style embedding part and affects only the transformer model.

Learning the style embedding. We propose a style embedding module to learn the general style representation. When the text is represented as a compressed dimension $z \in R^d$, the style information and content information are hard to separate. Therefore, we do not disentangle the latent representation z into style and content. Instead, we train the common representation depending on the style, and this common expression becomes style embedding. The set of style embeddings is S=$\{S_1, \cdots, S_k\} \in R^{d \times k}$, where k is the number of styles. The style embedding module uses a style classifier. The classifier



(a) Auto-encoder with added latent space

(b) Learning style embedding

Figure 2: The architecture of the two modules within our proposed model. The input sentence is represented as x, and s is the style label of the input. z^* denotes the combined latent representation, and 'sim' means similarity. In the image, the flow of the gradient is marked with red arrows. **(a)** The sentence generation model. The encoder has a sentence input x and generates a compressed expression z. The style embedding from the input text style s is added to this representation. Using this added latent representation, the decoder generates the reconstructed input. **(b)** The style embedding is obtained by the similarity between the input style and the latent representation of the input. The similarity is calculated by the dot product. Based on this similarity, the classifier predicts the style, and the ground truth is the input style label.

consists of a linear projection layer that calculates the probabilities $W \in R^{k\times 1}$ corresponding to each style. The input of the style classifier is the similarity between the latent representation z and style embeddings S. The similarities are calculated by the dot product, $sim_{z,S} = \{sim_{z,S_1}, \cdots, sim_{z,S_k}\}$. Based on the calculated similarities, the style classifier predicts the style label. Hence, the objective of the training is to make similarity constituting the ground truth of the latent representation achieves the highest value among all the calculated similarities. By making the similarity of the input style the highest value, the corresponding style embedding obtains a more proper representation of the given input style. For instance, if there are only two styles, such as positive and negative, the two styles are labeled 0 and 1; then, the style classifier contains the sigmoid function. This makes the negative style embedding a negative value representation. Similarly, the positive style embedding obtains a positive value expression. The classifier and the style embedding are trained by the classification loss L_{se} as follows:

$$L_{se}(C_{\theta_c}(sim_{z,S_i}, s_i)) = -\sum_{i=1}^{k} s_i \log(s_i). \tag{2}$$

where C_{θ_c} indicates the style classifier in the style embedding model, s_i means the input text's style label and sim_{z,s_i} represents the similarity between input's latent representation and the style label. The back-propagation procedure with respect to the classification loss does not affect the autoencoder model and latent space z, only the style embedding result.

Combining the latent space and style embedding. We finally modify the latent representation from the encoder by adding the learned style embedding. The modified latent representation, which includes information of the original text and style, becomes the input of decoder D. The added latent representation is expressed as follows:

$$z_x^* = z_x + w{\cdot}s_i. \tag{3}$$

The hyperparameter w reflecting the style strength modulates how much of the style will be changed in the sentence. In addition to the encoder output, the style embedding can be used to adjust the style part in the sentence. During training, the value of the strength is only 1, making the input representation slightly more inclusive of its own style. At this time, only the style of the input sentence is used, not other styles. With this added latent space z_x^*, the decoder reconstructs the input x as follows:

$$D_\theta(z_x^*) = \hat{x}. \tag{4}$$

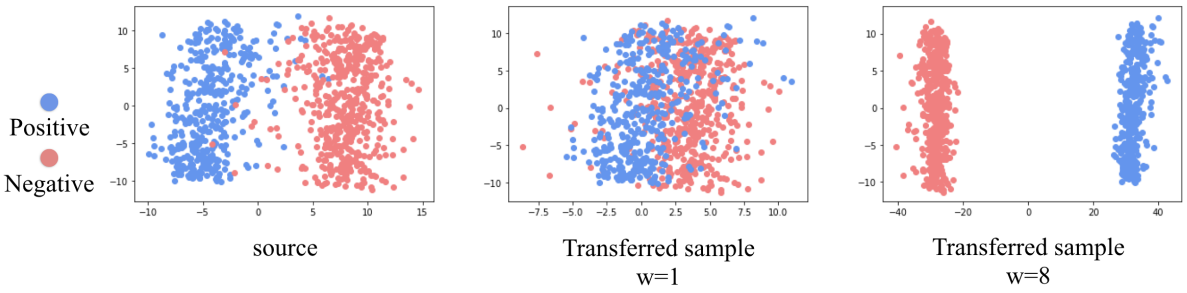


Figure 3: Visualization of representations with different style weights w. The direct output of the encoder is represented as source, and the one after adding the style embedding with the style strength w is indicated as a transferred sample. PCA is used to project the vectors into two-dimensional space. In the image of the source data, the left side of the projected space represents a positive style, while the right side implies a negative style according to the source image. As the weight increases, we can observe that the original negative samples move toward the positive position.

This approach slightly adds style information but does not significantly affect the reconstruction. In fact, the input sentence is reconstructed by the decoder if additional style information is not used. As is evident, the reconstruction model and the style model focus only on their tasks. Owing to this architecture, the style expression used inside the generation model can be adapted without any other constraints. At the test time, by adjusting the value of w, we can generate different output style strengths as desired. The larger the value of w is, the more the text style changes.

4 Experiments

4.1 Implementation

In our transformer-based sequence-to-sequence model, the latent size and dimension of self-attention are both 256. Additionally, the word embedding size and style embedding size are both 256. The size of the feedforward network (FFN) in the transformer is set to 1024. Each encoder is repeated two layers of the transformer, and the decoder's setting is the same. The smoothing parameter ϵ is 0.1. The style classifier consists of a linear layer, and the size of W is 2×1 because the number of styles is 2 (negative and positive). We use the Adam (Kingma and Ba, 2015) optimizer with an initial learning rate of 0.001. In the inference phase, we use the hyperparameter w, the values of which depend on the dataset. For the Yelp dataset, the set is $W = \{8.0, 9.0, 10.0\}$; for the Amazon dataset, the set is $W = \{9.0, 10.0, 11.0\}$

4.2 Datasets

Yelp. This dataset consists of restaurant reviews from the Yelp dataset; we use the dataset from Li et al. (2018). Each review is rated on a scale of 5 stars. A 2-star or 1-star review is considered a negative style, whereas a 4-star or 5-star review is considered a positive style. The total number of training samples is 443,259; the number of positive data is 266,041, and the number of negative data is 177,218. There are 2000 validation samples and 500 test samples for each style.

Amazon. The Amazon dataset comprises Amazon product reviews. We use the dataset from He and McAuley (2016). These data consist of two styles: negative and positive. The style labeling policy is the same as in the Yelp dataset. The number of training samples is 554,997; the number of positive samples is 273,150, and the number of negative sample is 272,919. Additionally, this dataset contains 1,015 samples for negative validation data and 985 for positive validation data. The number of test data samples is 500 for each style.

4.3 Evaluation protocol

We compare the performance of our model with the models of prior works based on the following three criteria. There is a trade-off between transferring style and preserving content. Moreover, the higher the

Table 1: Evaluation results on the Yelp test dataset and the Amazon dataset. The bilingual evaluation understudy (BLEU) score is computed by reference data that have been manually written by humans. Our model shows different results corresponding to the style strength w. The settings of our models are as follows: (1) the style strength is 10 for Yelp and 11 for Amazon; (2) the style strength is 9 for Yelp and 10 for Amazon; (3) the style strength is 9 for Yelp and 8 for Amazon.

Model	**Yelp**			**Amazon**		
	Accuracy	**BLEU**	**PPL**	**Accuracy**	**BLEU**	**PPL**
Human reference	69.4	100.0	62.7	46.3	100.0	75.8
Cross-Align (Shen et al., 2017)	75.4	6.0	45.4	56.0	1.0	34.3
MultiDec (Fu et al., 2018)	52.0	11.3	90.1	67.3	9.1	60.3
StyleEmb (Fu et al., 2018)	18.0	16.7	56.1	43.6	15.1	60.1
DeleteAndRetrieve (Li et al., 2018)	79.0	16.0	66.0	51.2	29.3	55.4
G-GST (Sudhakar et al., 2019)	73.6	25.9	49.7	31.3	29.9	42.6
B-GST (Sudhakar et al., 2019)	64.3	22.6	59.4	35.2	26.6	127.5
Style transformer (Dai et al., 2019)	77.1	29.2	42.4	40.1	41.1	42.2
Ours (1)	91.8	13.7	88.4	68.5	18.6	96.4
Ours (2)	86.9	15.2	80.8	62.4	21.0	88.7
Ours (3)	79.8	17.2	74.5	55.8	23.0	82.7

accuracy is, the lower the bilingual evaluation understudy (BLEU) score.

Reflecting the given style. We measure how much the style changes by the model using the fastText classifier (Joulin et al., 2017). We train this classifier on each training dataset; the number of training epochs is 25, a bi-gram sequence is used, and the learning rate is 1.0. The style control score for this criterion is the accuracy, and the higher the value is, the better the performance.

Content preservation. Content preservation is measured by the BLEU score (Papineni et al., 2002), which is a measure of how much the output sentence overlaps with the input sentence. The generated output sentence is compared with reference data written by a human. A higher BLEU score indicates that the transferred sentence preserves the source content.

Fluency. We use a language model toolkit to measure the fluency of the generated sentences (Stolcke, 2002). This language model estimates the prior probabilities of word strings and computes the perplexity (PPL) of each test sentence. We use tri-grams to train this model. This measurement is applied to the test data to obtain the PPL, where a smaller PPL value means a more fluent sentence.

4.4 Results

Quantitative evaluation result. Table 1 shows the Quantitative evaluation results on the Yelp and Amazon datasets. In the Yelp dataset, compared with the DeleteAndRetrieve model (Li et al., 2018), the model with the highest accuracy among the previous studies, our model with setting (1) shows a higher accuracy, while our model with setting (2) shows a better accuracy and BLEU score. Overall, the accuracy of our model is higher than that of the others, and the PPL performance is lower. In the Amazon dataset, our model's accuracy is slightly lower than that of the MultiDec model (Fu et al., 2018), but the BLEU score is much better. Additionally, the other models are able to generate only one sentence (single modality), but our models can generate multiple outputs from a single input by adjusting the strength. By altering the strength, trade-offs are induced between the accuracy and BLEU score. By decreasing the hyperparameter w, the style accuracy is decreased, and the BLEU and PPL scores are improved.

Adjusting the style strength. As mentioned before, our model can control how much the style of the sentence is changed by learning style embedding. In Figure 3, we use PCA (Wold et al., 1987) to visualize how the data embedding is changed as adjusting the style strength. More specifically, the latent representation that is the direct output of the encoder is represented as source, and the one after adding

Table 2: Generated samples corresponding to different controlled values of the hyperparameter w on the Yelp dataset. The greater the strength, the more it affects the style of the sentence.

Negative → Positive	
Input	other than that , food here is pretty gross .
w=8	other than that , food here is pretty **nice** and **authentic** salad .
w=9	other than that , food here is pretty **good** and **enjoy warm** .
w=10	other than that , food here is pretty **nice** and **authentic** salad .
Input	so , no treatment and no medication to help me deal with my condition .
w=8	so , **best** treatment and no medication to help me deal with my condition .
w=9	so , **best** treatment and no medication to help me deal with my life .
w=10	so , **best** treatment and **great** medication to help me deal with my life .
Positive → Negative	
Input	he is very thorough and genuinely cares for his customers .
w=8	he is very thorough and genuinely cares for his customers total saying .
w=9	he is very thorough and **never** cares for standing his customers under .
w=10	he is very **lazy** and did **not** cares for his customers mostly .
Input	this course is one of the finest in the area .
w=8	this course is one of the finest in the **trash** area .
w=9	this course is **not** one of the finest in the area pay .
w=10	this course is **not** one of the finest in the **trash** area .

the style embedding using the style strength w is indicated as a transferred sample. In the source data of Figure 3, the positive sentences are located on the left side, and the negative sentences are on the right side. As the degree of style increases, the positive samples move toward the right side, which denotes a negative style, while the negative samples move toward the left side of the image. In other words, as the style strength increases, the samples obviously transfer to different styles. Table 2 shows the generated outputs with different weight settings. As the transfer results vary, the sentence preservation performance is not good. There is a trade-off between the diversity of the results and the preservation of sentences. Additionally, Table 3 presents the outputs using their own style. For example, if the input is positive, the style inserted is also positive. The other method (Dai et al., 2019) shows the reconstructed outputs. However, the results of our model reflect the original style more strongly, and expressions such as "trashy", "barely", and "great" are added.

Separation of the generation part and style part. As mentioned before, the generation models employed in previous studies were responsible for both the sentence reconstruction and the style modification tasks. In our study, we separate the style part from the reconstruction model so that each model can focus on its own task. Table 3 shows the sentence reconstruction performance without using style information to verify that each component is effectively separated in our model. The previous model clearly does not use any style information, and the style changes even when the reconstruction is performed. However, our model completely focuses on the restoration of the input. As a result, the two parts of our model are well divided, and each part focuses on its own task.

5 Conclusion

In this paper, we proposed a transformer-based autoencoder model for the text style transfer task that is combined with a separate style embedding module based on the similarity measurement with the latent text representation. With this style embedding part, our model can adjust the style strength of the output. In both Yelp and Amazon datasets, our model showed better accuracy and BLEU scores than previous models. In particular, our approach generated a variety of outputs with regard to the style strength. Additionally, with the input's own style, the generated sentences could represent the initial style more

Table 3: Generated samples with their own style as the target on the Yelp dataset. The other model reconstructs only the input, while our model generates more sentimental sentences.

Negative → Negative	
Input	so , no treatment and no medication to help me deal with my condition .
(Dai et al., 2019)	so , no treatment and no medication to help me deal with my condition .
Ours	so , no treatment and no medication to help me deal with my **pain** .
Input	food was cold (still frozen) , i had the ribs .
(Dai et al., 2019)	food was cold (still frozen) , i had the ribs .
Ours	food was cold (still frozen) , i had the **trash** ribs .
Input	the pizza is offered without toppings and it 's lacking in flavor .
(Dai et al., 2019)	the pizza is offered without toppings and it 's lacking in flavor .
Ours	the pizza is offered without toppings and it 's **barely** lacking in flavor .
Positive → Positive	
Input	i love this place , the service is always great !
(Dai et al., 2019)	i love this place , the service is always great !
Ours	i love this place , the service is always great and **delicious** !
Input	the service was amazing and the staff was very friendly .
(Dai et al., 2019)	the service was amazing and the staff was very friendly .
Ours	the service was amazing and the staff was very friendly and **great** .
Input	it is the most authentic thai in the valley .
(Dai et al., 2019)	it is the most authentic thai in the valley .
Ours	it is the most authentic thai in the valley and **great service** .

Table 4: Reconstruction results without using style information. A lower accuracy means a better reconstruction performance because the accuracy score represents how much of the style is transferred. The Self-BLEU score is computed by the input test data.

Model	**Accuracy**	**Self-BLEU**	**PPL**
Style transformer (Dai et al., 2019)	43.1	58.8	45.0
Ours	7.0	65.1	48.9

strongly than the previous models. On the other hand, our model's PPL performance was lower than that of other methods. Our model focuses more on the reconstruction task than other models and this restoration causes a decrease in PPL performance because our model tends to restore content words such as conjunctions or connection words that need to be changed when the style changes. In the future, we would improve the model to address this issue. We also plan to apply the normalization schemes from the image domain, such as batch-instance normalization (Nam and Kim, 2018) and spatially adaptive normalization (Park et al., 2019), to our model. Exploring methods to combine style information with the latent space will be helpful for enhancing the performance.

Acknowledgements

This research was supported by the National Research Foundation of Korea grant funded by the Korea government (MSIT) (No. NRF-2019R1A2C1006608).

References

Ning Dai, Jianze Liang, Xipeng Qiu, and Xuan-Jing 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.

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

Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web*, pages 507–517.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1587–1596. JMLR. org.

Harsh Jhamtani, Varun Gangal, Eduard Hovy, and Eric Nyberg. 2017. Shakespearizing modern language using copy-enriched sequence to sequence models. In *Proceedings of the Workshop on Stylistic Variation*, pages 10–19.

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.

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

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. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. In *The International Conference on Learning Representations*.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Unsupervised machine translation using monolingual corpora only. *The International Conference on Learning Representations*.

Guillaume Lample, Sandeep Subramanian, Eric Smith, Ludovic Denoyer, Marc'Aurelio Ranzato, and Y-Lan Boureau. 2019. Multiple-attribute text rewriting. *The 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.

Dianqi Li, Yizhe Zhang, Zhe Gan, Yu Cheng, Chris Brockett, Ming-Ting Sun, and Bill Dolan. 2019. Domain adaptive 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 3295–3304.

Hyeonseob Nam and Hyo-Eun Kim. 2018. Batch-instance normalization for adaptively style-invariant neural networks. In *Advances in Neural Information Processing Systems*, pages 2558–2567.

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.

Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. 2019. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2337–2346.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Advances in neural information processing systems*, pages 6830–6841.

Andreas Stolcke. 2002. Srilm-an extensible language modeling toolkit. In *7th International Conference on Spoken Language Processing*, pages 901–904.

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 3260–3270.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *The IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, pages 2818–2826.

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.

Guoyin Wang, Chunyuan Li, Wenlin Wang, Yizhe Zhang, Dinghan Shen, Xinyuan Zhang, Ricardo Henao, and Lawrence Carin. 2018. Joint embedding of words and labels for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2321–2331.

Svante Wold, Kim Esbensen, and Paul Geladi. 1987. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52.

Jingjing Xu, Xu Sun, Qi Zeng, Xuancheng Ren, Xiaodong Zhang, Houfeng Wang, and Wenjie Li. 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)*, pages 979–988.

Yanpeng Zhao, Victoria W. Bi, Deng Cai, Xiaojiang Liu, Kewei Tu, and Shuming Shi. 2018. Language style transfer from non-parallel text with arbitrary styles. *The International Conference for Learning Representations*.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *The IEEE International Conference on Computer Vision(ICCV)*, pages 2223–2232.