Exploring Text Style Transfer Between Barack Obama and Donald Trump

Ulus Emir Aslan
Institute of Computer and Informatics
Istanbul Technical University
Istanbul, Turkiye
150210320
aslanu21@itu.edu.tr

Kaan Yücel
Institute of Computer and Informatics
Istanbul Technical University
Istanbul, Turkey
150210318
yucelk21@itu.edu.tr

***Abstract*—Text style transfer is about changing the stylistic attributes of text while preserving its original content. This paper explores the application of VT-STOWER (with adjustments) a variational autoencoder-based model which has great success for the tasks : sentiment style transfer, formality style transfer, and code switch style transfer, and also tries to create a specified autoencoder model with optimization to transfer text styles between Barack Obama and Donald Trump. Despite implementing advanced modeling techniques and style-aware training mechanisms, the approach faced challenges in achieving successful style transfer. We provide an analysis of the model's performance, discuss the natural difficulties in this task and suggest future possible implementations.**

***Index Terms*—Text style transfer, VT-STOWER, variational autoencoder, style embeddings, natural language processing, latent space**

I. Introduction

Text style transfer aims to transform text to a target style while preserving the original content. In this study, we investigate style transfer between the distinct textual styles of Barack Obama and Donald Trump, characterized by contrasting tones and different use of vocabularies. Primarily,we employ VT-STOWER, a model which uses style embeddings and sequential processing, to perform this task. Secondly, we employ autoencoder model with transformer blocks, style classifier at latent space level and several extra architecture preferences. However, achieving good style transfer remains challenging because of the complex relations between content and style of the personas.

VT-STOWER, initially introduced by Xu et al. [1], demonstrated its effectiveness in using pivot words to increase style transfer performance. The model combines variational autoencoders (VAEs) with style embeddings and attention mechanisms to capture style-specific features while maintaining content consistency. In their work, the model was evaluated on datasets with well-defined styles, such as Yelp for sentiment transfer and GYAFC for formality transfer, and achieved notable results. Building on this framework, our study applies VT-STOWER to a real-world context involving political speeches, where the distinct styles of Barack Obama and Donald Trump offer a unique and difficult challenge for text style transfer.

While the model effectively utilizes style embeddings to align latent representations with specific styles, transferring stylistic nuances from one domain to another—particularly in complex, real-world contexts—remains a significant challenge. This work explores these limitations and evaluates VT-STOWER's ability to handle subtle and overlapping stylistic features between these two distinct personas.

Our contributions are as follows:

- We adapt and evaluate VT-STOWER for text style transfer between the styles of Barack Obama and Donald Trump.
- We discuss the limitations of current style embedding methods and propose potential directions for improvement.

Dataset is created using the speeches of Barack Obama and Donald Trump. Data is taken from the open source website Miller Center Speeches are retrieved by using various web-scraping techniques using the Python library 'BeautifulSoup'. More information about the dataset is given in the later part of the paper.

II. Methodology

A. VT-STOWER Model Architecture

We use VT-STOWER, a model based on a variational autoencoder (VAE) framework with the following components and adjustments:

- **Token Embedding Layer**: Maps input tokens to high-dimensional embeddings. The tokens are generated with GPT2 tokenizer
- **GRU Encoder**: Processes sequential input and captures contextual information. In the original VT-STOWER model, this part is replaced with pretrained RoBERTa model.
- **Latent Space**: Encodes last hidden state of encoder GRU into a latent vector z using learned mean (μ) and variance (σ^2) with fully connected layers. Pairwise addition is applied between the latent vector and the style embedding of the source text in the **training**. And in the **inference** time, a pairwise addition is applied between the latent

vector and the difference between the target style and the source style.

- **Style Embeddings**: Represents stylistic attributes (Obama: 0, Trump: 1) in the latent space.
- **GRU Decoder**: Generates reconstructed text by processing the input sequence embeddings combined using pairwise summation with the latent vector. In the original VT-STOWER model, this part is replaced by a transformer model.

1) Training Objective: The training objective includes three main components and two steps:

- **Reconstruction Loss**: Ensures the decoded text retains the original content.
- **KL Divergence Loss**: Regularizes the latent space for smooth sampling.
- **Style Similarity Loss**: Aligns the latent representation with the desired style embedding.

2) Training Step 1: In the first training step, the model is trained to reconstruct the input text while simultaneously learning to align the latent representation (z) with the corresponding style embedding (s_i). This step involves training the style embeddings as well as the latent space.

The training objectives include:

- **Reconstruction Loss** ($\mathcal{L}_{\text{recon}}$): Ensures that the decoded text closely matches the original input. This is calculated as:

$$\mathcal{L}_{\text{recon}} = \text{CrossEntropy}(x, \hat{x})$$

where x is the input sequence and $\hat{x}$ is the probabilities of tokens (2D list).

- **KL Divergence Loss** ($\mathcal{L}_{\text{KL}}$): Regularizes the latent space for smooth sampling. This is calculated as:

$$\mathcal{L}_{\text{KL}} = -0.5 \cdot \sum_{j=1}^{\dim(z)} \left(1 + \log(\sigma_j^2) - \mu_j^2 - \sigma_j^2\right)$$

- **Style Similarity Loss** ($\mathcal{L}_{\text{style}}$): Aligns the latent representation with the correct style embedding (s_i) using cosine similarity:

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

Here, $d_i = 1$ for the source style of the input, and $d_i = 0$ for all other styles. The stop-gradient operation ($\text{sg}(\cdot)$) prevents updates to z during this step.

This step trains the entire model, including the style embeddings. The optimization objective combines the three losses as:

$$\mathcal{L}_{\text{step1}} = \lambda_{\text{vae}} \cdot \mathcal{L}_{\text{vae}} + \lambda_{\text{style}} \cdot \mathcal{L}_{\text{style}}$$

where λ_{vae} and λ_{style} are hyperparameters controlling the weight of the reconstruction, KL and style similarity losses.

3) Training Step 2: In the second training step, the model is trained with a focus on pivot words to increase its ability to transfer style while preserving content. Here, the style embeddings are frozen, and the pivot words are identified based on their importance scores. These scores are computed using:

$$\begin{aligned} \text{GRU Attention Scores:} \quad & \alpha_t = \text{softmax}\left(\cos\left(h_t^\top, h_{\text{final}}\right)\right) \\ \text{Style Similarity Scores:} \quad & \beta_t = \text{softmax}\left(\cos\left(e_t, s_{\text{src}}\right)\right) \\ \text{Combined Scores:} \quad & \gamma_t = \frac{\alpha_t + \beta_t}{2} \end{aligned}$$

Where h_t is the output hidden state of the encoder GRU after processing the embedded word e_t, s_src is the source style. Pivot words are selected based on the top scoring tokens. These tokens are then masked in the input sequence by replacing them with a learnable `<mask>` token. This implementation differs from the original VT-STOWER paper since we do not use transformers as encoder and decoders

The training objectives in this step are:

- **Reconstruction Loss** ($\mathcal{L}_{\text{recon}}$): Encourages the model to reconstruct the original sentence while accounting for the masked pivot words.
- **KL Divergence Loss** ($\mathcal{L}_{\text{KL}}$): Regularizes the latent space, as in Training Step 1.

The training objective for this step is:

$$\mathcal{L}_{\text{step2}} = \mathcal{L}_{\text{recon}} + \beta \cdot \mathcal{L}_{\text{KL}}$$

In this step, style embeddings are no longer updated, ensuring that the learned styles remain stable. The masked pivot words force the model to rely more heavily on the latent style representation for reconstruction, thereby enhancing style transfer capabilities.

4) Pivot Masking: To enforce style-focused training, we mask pivot words (style-indicative terms) during training. Pivot masking is guided together by attention scores derived from the encoder's last hidden state and its former hidden states and the alignment of token embedding with the style embedding, which highlights stylistically and contextually significant words.

B. Transformer Autoencoder Architecture with optimization

This model is added as an **extra option** for the future since we did not manage to function this model properly for now. We get inspired by the paper [3] and implemented some the architecture styles and optimization techniqiues from them. This Transformer Autoencoder model's core idea is to capture complex dependencies within the input sequence and generate meaningful representations of the data.Then using style classifier and optimization try to change the latent space representation to affect style. Below is a detailed description of the architecture:

1) Input Embedding Layer: Input sequence is represented by a vector in a continuous space. The embedding process transforms the inputs with BERT tokenizer.

2) Positional Encoding: : Positional encoding is added to the embeddings to introduce information about the position of tokens within the sequence.

3) Encoder: The encoder is composed of multiple transformer blocks. Each transformer block has linear layers and multihead attention mechanism with layerNorm and dropout techniques.

a) Multi-Head Self-Attention: The self-attention mechanism is crucial as it allows the model to learn relationships between tokens in the sequence regardless of their distance from each other. Each token attends to all other tokens in the sequence, capturing long-range dependencies.

b) Feed-Forward Neural Network (FFN): After the attention layer, the output is passed through a position-wise feed-forward network, which consists of two fully connected layers with a ReLU activation function between them. This layer further learns the features after the attention extracts the information.

c) Layer Normalization and Residual Connections: The residual connection and layer normalization allows gradients to flow more easily during training, handling the vanishing gradient problem and improving convergence.

4) Latent Space: The encoder compresses the input sequence into a lower-dimensional latent representation with fully connected layer.

5) GRU layer: Lower dimensional form passed through the GRU layer to further learn the relationship between latent space dimensions.

6) Decoder: The decoder in the Transformer Autoencoder reconstructs the input sequence from the latent representation provided by the encoder. The decoder is also made up of multiple identical layers and shares many components with the encoder.

7) Loss Functions: In this section, we discuss two important loss functions used in the model: the *Attribute Loss* and the *Reconstruction Loss*. These loss functions play key roles in optimizing the model during training.

The *Attribute Loss* focuses on measuring the accuracy of predicted attribute probabilities, while the *Reconstruction Loss* is used for sequence prediction tasks, incorporating label smoothing to prevent overfitting and improve generalization. Both losses help optimize different aspects of the model's performance and are key components of the overall training objective.

8) Fast Gradient Iterative Optimization: This method is an optimization technique used to iteratively modify an input tensor z in order to cause the classifier to make incorrect predictions. This method can be used to generate various examples. Below is a detailed explanation of the procedure used in this method.

a) Method Overview: Given an input tensor z and a classifier classifier(z), the goal is to modify z such that the output of the classifier is close to the target label y_{prime}. The modification process uses the following key components: - $\mathbf{w_i}$: The weight for each gradient update step. - decay_coeff: A decay factor for reducing the weight at each step. - threshold: A value to stop the process if the prediction is sufficiently close to the target. - s_steps: The number of steps to perform for each weight.

The procedure iterates over multiple weight values and performs gradient-based updates to the input z, aiming to decrease the loss between the predicted output and the target.

b) Mathematical Formulation: The procedure operates as follows:

1) Start by cloning the input tensor z to a new tensor $z_\star$, which will be modified.
2) For each weight w_i in the list of weights, perform the following steps:
   a) For each step in the range of s_{steps}, compute the gradient of the loss function with respect to $z_\star$.
   b) Compute the classifier output $\hat{y} = \text{classifier}(z_\star)$ and the binary cross-entropy loss:
   $$L(z_\star) = -y_{\text{prime}} \log(\hat{y}) - (1 - y_{\text{prime}}) \log(1 - \hat{y})$$
   where y_{prime} is the target label.
   c) Backpropagate the loss to compute the gradient of the loss with respect to $z_\star$, i.e., $\nabla_{z_\star} L(z_\star)$.
   d) Update $z_\star$ using the gradient and the weight w_i:
   $$z_\star \leftarrow z_\star - w_i \cdot \nabla_{z_\star} L(z_\star)$$
   e) Compute the predicted class $\hat{y}_{\text{new}} = \sigma(\text{classifier}(z_\star))$ using the sigmoid activation function, where σ is the sigmoid function.
   f) If the mean absolute difference between the target label y_{prime} and the predicted label $\hat{y}_{\text{new}}$ is smaller than a given threshold, stop the process and return the modified $z_\star$.
3) At each step, reduce the weight w_i by multiplying it by the decay coefficient decay_coeff to gradually reduce the update magnitude.
4) If the threshold is not reached, the function returns the modified input $z_\star$ after all iterations are completed.

c) Optimization Process: The method can be viewed as an optimization process where we aim to modify the input z to minimize the loss with respect to the target output. The key elements of this method include: - Gradient-based updates to adjust $z_\star$ iteratively. - Use of binary cross-entropy loss for binary classification tasks. - A weight schedule that reduces the magnitude of updates over time using the decay coefficient. - An early stopping criterion based on the distance between the predicted output and the target label.

This method allows for efficient adversarial example generation, as it iteratively adjusts the input based on the gradients of the loss function with respect to the input.

C. Application

Unfortunately, we could not properly function this optimization technique currently because of the autoencoder does not give decent results. However, we aim to make this model better and apply the fast iterative gradient optimization technique to achieve more healthy outputs in the future.

III. Real World Applications

A. Potential Use Cases

- Political communication and speech rephrasing.
- Creative text generation and content rewriting.
- Analyzing stylistic differences.
- Character analysis

IV. Experimental Evaluation

A. Dataset

We use a dataset comprising speeches from Obama and Trump. Texts are preprocessed and cleaned, tokenized using tokenizer and labeled according to their source (Obama or Trump). We get this data using web scraping techniques. Data set containts approximately 400.000 words for Trump and 350.000 words for Obama.

B. Evaluation Metrics

- **Reconstruction Loss**: Measures content preservation Cross Entropy loss.
- **Style Transfer Success**: Assesses style embedding alignment via cosine similarity.
- **Qualitative Analysis**: Includes examples of generated text.

C. Results and Analysis

The model achieved partial success in reconstructing the input sentence and differ the style embeddings but failed to generate consistently stylized text. Key findings include:

- The style similarity loss plateaued around 0.68, showing incomplete alignment between latent vector z and style embeddings.
- Low cosine similarity between 2 style embeddings (0.01) which shows model can capture differences of Obama and Trump speeches.
- Example outputs revealed that content was often preserved, but stylistic transformation was insufficient. Only 2-3 words change and it does not suit the content much.

V. Conclusion

A. Summary

This study highlights the challenges of text style transfer between Obama and Trump using VT-STOWER. While the model captured some stylistic features, achieving effective transfer remains difficult.

B. Future Work

Future directions include:

- Using advanced architectures (e.g., Transformers).
- Enhancing style embeddings with additional supervision and inductive biases.
- Trying different approaches rather than pivot masking and VT-STOWER.
- Trying to better the transformer autoencoder model with fast iterative gradient optimization technique

Acknowledgment

Thank you Faik Boray Tek, Atakan Site.

References

[1] H. Xu *et al.*, "VAE Based Text Style Transfer with Pivot Words Enhancement Learning," *arXiv preprint arXiv:2112.03154*, 2021.

[2] F. Locatello *et al.*, "Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations," in *Proceedings of the 36th International Conference on Machine Learning*, vol. 97, pp. 4114–4124, 2019.

[3] K. Wang, H. Hua, and X. Wan, "Controllable unsupervised text attribute transfer via editing entangled latent representation," in *Advances in Neural Information Processing Systems*, vol. 32, 2019.