CS230

Style Transfer For Poetry

Anthony Maltsev
Department of Computer Science
Stanford University
amaltsev@stanford.edu

Michael Young-jin Cho
Department of Computer Science
Stanford University
mycho21@stanford.edu

Abstract

Textual style transfer is the problem of preserving meaning while changing textual style. This has a wide variety of applications, ranging from language translation to text simplification or debiasing offensive texts. In this project, we investigate translating authorship for poetry, specifically translating works by Shakespeare into the style of New York School poets like Frank O'Hara and vice versa. We implement a modified version of the Iterative Back Translation technique introduced by Hoang et al in 2018 [1], which involves training two models that produce synthetic parallel training data for each other in an iterative training loop.

1 Introduction

Neural style transfer in natural language processing is a task that seeks to preserve the semantic content of an input text while transforming it into a different style. This has a lot of useful applications in the real world: helping people write things by suggesting ways to make their writing more formal in a professional setting; rewriting complicated ideas in a simpler style to make text more accessible to a broad audience; translating news articles for publication in countries that speak a different language; protecting the anonymity of authors of controversial texts by converting their writing to a neutral style as people can often identify a writer by their style; bias mitigation to rewrite biased or discriminatory text without said bias; or artistic experimentation. This broad range of applications makes style transfer for natural language an attractive and burgeoning field of research.

Neural style transfer for images is a familiar application where given a style image and a content image, you wish to generate an output image that has the same visual style as the style image while preserving the content of the content image. This is an unsupervised optimization task, where a model is able to generate data without examples of style/content inputs and output pairs (i.e. parallel data). This allows these models to be fairly easy to use since any arbitrary image data (which is widely available) can be used.

One of the primary difference between textual style transfer and visual style transfer is that, traditionally, textual style transfer relied on having a corpus of parallel data. For example, an English $\leftrightarrow$ German translation model might be trained on news data that was manually translated from one language into the other. However, there is no such corpus of parallel data widely available for many textual style transfer target applications. This prompts the development of training methods on two separate mono-style corpora.

A survey on deep learning for textual style transfer [2] identifies 3 primary approaches for training translation models on non-parallel, mono-style corpora: disentanglement, prototype editing, and pseudo-parallel corpus construction. In this project, we investigate the Iterative Back Translation (IBT) method of pseudo-parallel corpus construction introduced in 2018 by Hoang et al [1] on translating poetic text from one style (Shakespearean) to another (New York School).

CS230: Deep Learning, Fall 2024, Stanford University, CA.

2 Related work

We will now discuss other methods used for textual style transfer both on parallel and non parallel corpora before discussing the approach we chose to implement (Iterative Back Translation).

Most methods that tackle a text style transfer task with available parallel data use a standard neural sequence-to-sequence model with an encoder-decoder architecture. Historically, these would be LSTMs but in recent years, the transformer architecture has massively outshined LSTMs. There is active research in how to improve the basic sequence-to-sequence models for text style translation, including research on data augmentation for parallel data examples [11] and on inference-time techniques to improve translations [3].

Disentanglement is a technique for textual style transfer over non parallel data that consists of three steps: 1.) encode the text into a sufficiently expressive latent representation, 2.) manipulate the latent representation to remove the input style attribute and add back the target style attribute, and 3.) decode the resulting manipulated latent representation into a target text that preserves the content but has transfered style. This general technique can be used with a variety of encoder/decoder architectures and different choices for latent space manipulation (as well as loss functions for the manipulation that encourage content preservation while transfering style). Common choices for encoder/decoder are a Variational Auto Encoder (VAE) or a Generative Adversarial Network (GAN). There are three common choices for latent representation manipulation. First, Latent Representation Editing (LRE) uses a learned attribute classifier (such as an MLP) to iteratively update the latent encoding while ensuring content alignment with the input text. Second, Attribute Code Control (ACC) ensures the encoder removes all style information via an adversarial model and then the decoder is trained to add in the desired style. Third, Latent Representation Splitting (LRS) ensures that encoder disentangles the style from the content (ie some dimensions of the latent representation correspond to style and some to content) and then change the style attribute section of the latent representation with the desired style before decoding. State of the art using this method seems to be either this paper by Li et al [5] with uses a GAN and ACC or this paper by Yi et al [10] which uses a VAE and ACC.

Prototype editing is a technique for textual style transfer over non parallel data that does not use an end-to-end model like IBT or disentanglement. This technique consists of 3 steps: 1.) detect all style attribute markers in the input text and delete them, leaving a style-free content-only text, 2.) retrieve appropriate style attribute markers corresponding to the target style to replace the markers we deleted, and 3.) combine the content-only sentence with the style markers from step 2 to create a new, stylized sentence. Additionally, an extra step that checks whether the resulting sentence is fluent is often added. This technique is similar to older methods of textual style transfer which relied on simple word replacement according to precomputed translation dictionaries or hand-crafted translation rules. The first step of style marker detection is usually done by comparing n-gram frequencies between the two styles you are trying to translate, using attention based classifiers, or by using a mix of those two methods. The second step of candidate style marker generation is done by comparing similar content-only sentences between the two styles and comparing the style markers in their stylized versions. Generating the output sentence is generally done by using a pretrained language model that combines style markers and content. This model can be trained on the output of step one, which separates the style markers and content. Prototype editing has a huge benefit of being highly interpretable and also highly controllable as one can see which markers are placed where and how style is embedded into the sentence. This method was introduced by Sudhakar et al [7] in 2019 and remains the state of the art.

There has been previous work on translating Shakespearean language to a more modern style by Xu et al in 2012 [9]. This work relied on a parallel data corpus of Shakespearean plays and their manually translated copies in Modern English prose. This was an early attempt of automatic style transfer with a given target writing style. Their models were trained using a classical phrase-based statistical machine translation model, Moses introduced by Koehn et al in 2007 [4].

3 Dataset and Features

We are using the PoetryFoundationData dataset available on huggingface. We define our two monostyle corpora by taking the subsets of this dataset – one dataset of all the Shakespeare Poems,

and one dataset of all poems by prominent New York School poets (John Ashbery, Barbara Guest, James Schuyler, Kenneth Koch, and Frank O'Hara).

The dataset poses a major challenge to our project. One of the critical challenges that might prevent good results is that there is not a lot of data available for each of the two styles that we have chosen to translate – only on the order of 1000 training sentences in either of the two styles. This is especially challenging because authorship is considered to be a hard style as it is highly entangled with content (especially in the sphere of poetry). It will be interesting to see if our methods are able to overcome this challenge. We hope that pre-training a more general language model on a wider set of poetry training data (the PoetryFoundation dataset) will allow our models to do better despite the limited training data.

When beginning our project, we intended to train our own very rough language model based on the Poetry Foundation data. To do this, we processed all of the poems by splitting them into sentences, lowercasing all words, and getting rid of any non-alphabetic characters. Then, we use pre-trained GloVe word embeddings on each word to create a 50-dimensional feature vector for each word in each sentence sequence. We pad the data to 50 words, which is about twice the length of the average sentence. However, we were not able to train a model that would produce any good outputs, which meant we would struggle to evaluate our translation methods. We chose instead to use a pre-trained language model (a 220M parameter model based on Gemini) which we fine tuned to take text and output that same text back. This would ensure that our translator models initially output text that preserves semantic content, and style would be learned by the IBT training procedure.

4 Methods

Iterative Back Translation (IBT) is a method used to create a synthetic corpus of parallel data, which is significantly easier to train on than two separate mono-style corpora. This technique relies on using two translation models, $M_{A\to B}$ which maps elements of style A to their translation in style B and $M_{B\to A}$ which does the reverse. The system then trains these models collaboratively by iterating the following procedure:

1. Use the model to generate synthetic parallel corpora. Specifically, $M_{A\to B}$ will generate pairs (a, a') for each element $a \in A$ and $M_{B\to A}$ will generate pairs (b, b') for each element $b \in B$.
2. Train each model on their counterpart's outputs from step 1. In other words, train $M_{A\to B}$ on the dataset $\{(b', b) \quad \forall b \in B\}$ and train $M_{B\to A}$ on the dataset $\{(a', a) \quad \forall a \in A\}$

Following this procedure, as both of the models get better, they will produce better training data for each other, which will in turn allow them to train even better.

IBT relies on the assumption that, as you iterate, the training data produced by the models for each other actually gets better/more helpful for training. This is a strong assumption since there is no explicit constraint that ensures each iteration will produce better corpora. We address this problem by adding additional losses that encourage the translations to be better. We introduce a separately trained style classifier/discriminator that we will use to grade the style of translations. We finetuned OpenAI's GPT2 model to do text classification as the discriminator model. This addition of extra loss functions is an approach that was described by Zhang et al [12].

Additional loss terms that guide the model are crucial especially in the first few iterations of the system, since the initial synthetic parallel corpora are likely to be very bad. The addition a discriminator loss term to encourage good style transfer will help the model train at the beginning when the the model does not produce good synthetic data. This addresses one of the key problems with IBT identified by [2] – that the random initialization of the models might not produce good results because the training data produced by the feedback loop is too noisy to help training.

We further address this issue by not using a random initialization. We implement some transfer learning from a more general language model trained on a wider corpus of poetry data. We hope that initializing our training models with a more sophisticated language model that has been trained on a wider set of data will make the initial step of IBT less noisy and encourage faster convergence to a good translation model. We chose this initial model to be a generally trained language model (a 220M parameter model based on Gemini) that was finetuned on a wider poetry corpus. We chose to

train the language model to output the same sentence that was input into the model, since this would represent a translation that perfectly preserves semantic content, hoping that training with the IBT loop would learn style transfer, while preserving that semantic content.

We considered adding a cycle consistency loss term to the training loop in order to encourage the model to preserve semantic content during the translation. However, we chose to address this issue using an initialization procedure that encourages the model to preserve content, as described above. If we observed that the semantic content of translations was drifting too far from the original input, we believe that adding back this cycle consistency loss would encourage the model to preserve content.

The final overall loss function that we used for training was a weighted sum of the discriminator loss and the training loss from training the language model on the synthetic parallel corpus data. This introduces 2 new hyperparameters that control the weighting of these loss terms in the gradient computation.

We used the huggingface transformers [8] library to train the initial models based on GPT2 and Gemini as we described above. We implemented the Iterative Back Translation training loop using a custom gradient function in tensorflow [6].

5 Experiments/Results/Discussion

We trained two different types of models: a discriminator and the translators. For the discriminator, we achieved very good results in classifying whether a text was written by Shakespeare or by a more modern New York School poet. Our main metric of choice for this was accuracy. On a validation set of 100 text examples by either Shakespeare or a New York School poet, the discriminator achieved 100% accuracy. We did not heavily test this discriminator beyond this simple evaluation because it was not our primary objective with this project.

Our translation models unfortunately did not achieve any good results. We mostly evaluated our results by observing input/output pairs and qualitatively noting how well they maintained semantic content and how well they transferred style.

When we trained the two translators using IBT initialized as we described above (with a language model fine tuned to initially output the same text as was inputted), they were unable to break this initial fine-tuning. Even as we trained the models, they continued to output the same text as the input. For example:

```
input(NY): Elm Grove, Adcock Corner, Story Book Farm?

output(S): Elm Grove, Adcock Corner, Story Book Farm?
```

or

```
input(S): Here the anthem doth commence:  Love and constancy is dead;

output(NY): Here the anthem doth commence:  Love and constancy is dead;
```

We believe that this might be caused by overfitting during the fine-tuning step of the initial models. Also, we had an error in the code for the discriminator loss term which meant that there was no guidance from the discriminator to encourage it to break this pattern. The error is that the chain of differentiation is broken between generating outputs using the translator models and calculating the discriminator loss because the text generation (sampling from output probabilities) is not a differentiable function. Therefore, while we were able to compute a loss for the discriminator, it was not resulting in any gradients changing, so this term was not guiding the model. If we had more time for the project, we would change the discriminator that we had created to take the output logits before the text generation step as inputs in order to keep the chain of operations differentiable, but this was not feasable in the time we had to do the project.

We also trained the two translators using IBT initialized as sequence-to-sequence models that were not fine tuned at all (Google T5-small). These models were trained on a variety of sequence to sequence tasks, including language translation. We used this model to test the baseline IBT system as described in the paper without any additional loss terms to guide it. The results from this training were largely nonsensical as the training was not able to overcome the initialization noise of the system.

However, there were some very subtle signs that the model was learning something to do with the translation, as we can see in this generated New York/Shakespeare input/output pair:

```
input:  to the nose, a resinous thought, to the eye, a lacquered needle green...

          output:  a squeezed green, a resinous thought, to the eye
```

Here, the word squeezed is a novel generation that does fit in the style of Shakespeare (squeeze is in the Shakespeare vocab). However, this is likely not indicative of any real learning and is probably a product of noise because the vast majority of other outputs are either nonsense, or simply match examples that the original T5 model was trained on, such as English to German translation. Here are some examples:

```
input(NY): Bumped from our dog-perch, we'd had to roil with the last of them.

                  output(S): a few squirms, we'd had to ro
```

or

```
              input(S):Thou canst not see one winkle in my brow;

     output(NY): Du kannst nicht eine winkle in meinem braunen blicken;
```

We did not collect any quantitative evaluation of the translator models because we were not able to get them to produce any sensible output. Overall, our results confirm that the task of style transfer for text without proper parallel corpora is very challenging. We likely did not have enough data in our dataset to overcome the system noise.

Our code for the system can be found on github here: `https://github.com/anthonymaltsev/poetry-transfer`

6 Conclusion/Future Work

Iterative Back Translation is an interesting method for training a translation model without a well structured parallel corpus of training data. However, it is a very noisy training method, especially in its early stages so additional infrastructure to ensure that the models train effectively is necessary, whether this is initializing the models on fairly robust existing language models or adding additional losses that help guide the models in the direction that you want. We believe that our methods were not able to achieve good results because there is a lack of good training data. Despite our attempts to overcome the noise inherent to the training method, our lack of data meant that we were not able to train the models sufficiently. The next step in this work would be to fix the discriminator loss step to be differentiable in order to be able to use that to get useful gradients for the models.

In the future, it would be interesting to try these same methods on an easier translation task that has more available data and less entangled style attributes. It would also be interesting to consider additional or different loss terms that would guide the model training.

7 Contributions

- **Anthony Maltsev:** wrote the milestone and final report, wrote IBT training loop code, wrote some data preprocessing, finetuned discriminator model, debugged.
- **Michael Cho:** finetuned initial language models, handled AWS/cloud compute, debugged, tuned hyperparameters and trained final IBT model.

References

[1] Vu Cong Duy Hoang et al. "Iterative Back-Translation for Neural Machine Translation". In: *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*. Ed. by Alexandra Birch et al. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 18–24. DOI: 10.18653/v1/W18-2703. URL: https://aclanthology.org/W18-2703.

[2] Di Jin et al. "Deep Learning for Text Style Transfer: A Survey". In: *Computational Linguistics* 48.1 (Apr. 2022), pp. 155–205. ISSN: 0891-2017. DOI: 10.1162/coli_a_00426. eprint: https://direct.mit.edu/coli/article-pdf/48/1/155/2006608/coli_a_00426.pdf. URL: https://doi.org/10.1162/coli%5C_a%5C_00426.

[3] Tomoyuki Kajiwara. "Negative Lexically Constrained Decoding for Paraphrase Generation". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Ed. by Anna Korhonen, David Traum, and Lluís Màrquez. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 6047–6052. DOI: 10.18653/v1/P19-1607. URL: https://aclanthology.org/P19-1607.

[4] Philipp Koehn et al. "Moses: Open Source Toolkit for Statistical Machine Translation". In: *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*. Ed. by Sophia Ananiadou. Prague, Czech Republic: Association for Computational Linguistics, June 2007, pp. 177–180. URL: https://aclanthology.org/P07-2045.

[5] Yuan Li et al. "Complementary Auxiliary Classifiers for Label-Conditional Text Generation". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.05 (Apr. 2020), pp. 8303–8310. DOI: 10.1609/aaai.v34i05.6346. URL: https://ojs.aaai.org/index.php/AAAI/article/view/6346.

[6] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[7] Akhilesh Sudhakar, Bhargav Upadhyay, and Arjun Maheswaran. *Transforming Delete, Retrieve, Generate Approach for Controlled Text Style Transfer*. 2019. arXiv: 1908.09368 [cs.CL]. URL: https://arxiv.org/abs/1908.09368.

[8] Thomas Wolf et al. *HuggingFace's Transformers: State-of-the-art Natural Language Processing*. 2020. arXiv: 1910.03771 [cs.CL]. URL: https://arxiv.org/abs/1910.03771.

[9] Wei Xu et al. "Paraphrasing for Style". In: *Proceedings of COLING 2012*. Ed. by Martin Kay and Christian Boitet. Mumbai, India: The COLING 2012 Organizing Committee, Dec. 2012, pp. 2899–2914. URL: https://aclanthology.org/C12-1177.

[10] Xiaoyuan Yi et al. "Text Style Transfer via Learning Style Instance Supported Latent Space". In: *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*. Ed. by Christian Bessiere. Main track. International Joint Conferences on Artificial Intelligence Organization, July 2020, pp. 3801–3807. DOI: 10.24963/ijcai.2020/526. URL: https://doi.org/10.24963/ijcai.2020/526.

[11] Yi Zhang, Tao Ge, and Xu Sun. "Parallel Data Augmentation for Formality Style Transfer". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Ed. by Dan Jurafsky et al. Online: Association for Computational Linguistics, July 2020, pp. 3221–3228. DOI: 10.18653/v1/2020.acl-main.294. URL: https://aclanthology.org/2020.acl-main.294.

[12] Zhirui Zhang et al. *Style Transfer as Unsupervised Machine Translation*. 2018. arXiv: 1808.07894 [cs.CL]. URL: https://arxiv.org/abs/1808.07894.