Controllable Text to Text Style Transfer Between Genres

Mitry Anderson
mitryand@umich.edu

Abstract

What if you could translate a dry news article into the style of a classic adventure story? This project attempts to do just that, by exploring a method for style transfer that iteratively modifies a sentence's representation in the latent space between a transformer's encoder and decoder until the sentence achieves the desired attributes. This approach is extended to a genre style transfer text on a modified version of the Brown Corpus, that cuts the books in the corpus (which are sorted by genre) into paragraph sized chunks for training. The final performance could have been much better, as the model achieved low scores on the accuracy, fluency, and content preservation metrics that were used. While this approach remains promising, and future work may advance the topic further, it is also possible that genre is too abstract to achieve good style transfer using this method.

1 The Problem

Style Transfer in Natural Language processing is an attempt to generate a new piece of text, keeping the content of the input while changing its style to something new. Some common tasks include the transfer of sentiment from positive to negative, or negative to positive, translating biased language to neutral language, and translating between formal and informal writing patterns. There are a variety of approaches to achieve these different forms of style transfer. For this project I focused on the ability to control genre attributes in the latent space, because I thought it would be interesting to see how well this method of style transfer works on a more abstract task.

2 Related Work

Within the field of style transfer, the approaches used depend on the availability (or lack thereof) of parallel data (where there is a semantically identical version of the same sentence in each style)(Jin et al., 2020). Parallel data is difficult to come by, because it is uncommon for somebody to write the same sentence multiple times in different styles in natural language. For the task of genre style transfer, parallel data will be almost impossible to come by, because it would require many people to write the same books, articles, or other text in multiple genres. As such, I have investigated a few approaches to style transfer on non-parallel datasets. All of these involve using some form of autoencoder to learn an embedded representation of a sentence, which can then be used to asses and modify its style. The types of autoencoders used could make use of a variety of architectures, including recurrent neural networks and transformers. These autoencoder based approaches differ from other approaches that make use of Generative Adversarial Networks (GANs), which have found success for style transfer in computer vision (Jin et al., 2020). While some promising results have also been seen using GANs for linguistic style transfer, I have elected not to explore this route due to my lack of experience in the area.

Many approaches make use of an encoder/decoder architecture, and use the semantic information that is hopefully represented in the latent space between the encoder and decoder to capture information about the meaning and style of the sentence (Jin et al., 2020). One approach to formal/informal style transfer made use of a sequence-to-sequence network with a shared encoder for each style (to generate a shared latent space), but a separate decoder for each style (Wang et al., 2020). One issue with such an approach is that it will increase the model size to train multiple decoders for each attribute, and it will also be more difficult to transfer to other tasks.

A similar approach makes use of a shared encoder and decoder across all attributes, but with a classifier trained on the latent space to differentiate between attributes (Wang et al., 2019). In this ap-

proach, the translation to a different style is done by gradient descent in the latent space, with the gradient calculated based on the loss of the classifier with respect to the desired attributes. This approach achieved an accuracy of 85.3%, a perplexity of 47.4, and a BLEU score of 34.1 on the Amazon review sentiment transfer dataset. This type of model also has the added benefit of only needing to train a relatively smaller style classifier for each style attribute, rather than an entire decoder.

Another approach to this latent space modification uses an RNN based variational autoencoder to learn the latent representation, and then trains one classifier to predict a sentence's style, and another to predict its content (Liu et al., 2019). By training classifiers to predict both the style and content of a sentence, they are able to use a gradient descent based optimization as in Wang et. al, to attempt to achieve a better balance between style and content correctness, or a bias for one over the other. In one configuration, this approach achieved an accuracy of 90%, a perplexity of 15.9, and a BLEU score of 16.3 on the Amazon review sentiment transfer dataset. This apporach of minimizing both style and content loss mirrors similar approaches to style transfer that have seen success in the field of computer vision.

3 Approach

3.1 Data Set

The data set I used came from the Brown Corpus, downloadable through nltk (Bird et al., 2009). This give me access to samples of texts labeled with their genre, from 15 different genres. I partitioned 80% of the data for training, 18% for evaluation, and a small 2% for a final test set. The dataset was initially sorted into sentences. However, because it might be difficult to tell the genre of a text from a single sentence, I packaged the sentences into groups of 3, roughly equating to a short paragraph. For a given batch, the sentences needed to be padded to fit the maximum sentence length of 100 tokens, with remaining tokens being truncated.

This approach allows for relatively easy creation of a dataset of sentences sorted by genre, which is suited to the style transfer task that I am interested in. A major drawback is that it is not possible to re-write every text in the corpus in the style of each genre to create a comprehensive set of gold standard labels, which makes assessing performance more difficult. Additionally, some more detailed pre-processing to identify actual paragraph breaks could lead to increased performance in the future.

3.2 Model

Taking inspiration from the autoencoder based approaches described above, I implemented my own style transfer model (Wang et al., 2019), (Liu et al., 2019). A high level model diagram can be seen in Fig. 1.

Due to the recent success of transformers for language modeling, I decided to implement a transformer based auto-encoder to encode paragraphs into and decode paragraphs from the latent space (Wang et al., 2019). Rather than implementing this from scratch, I used use the cased version of the BERT Language model, loaded using the Hugging Face API for both the encoder and decoder of an EncoderDecoder model (Radford et al., 2019). I decided to use BERT because it was initially trained to be an auto-encoding language model, which seemed to fit with my intended use case quite nicely (Devlin et al., 2018). This model was fine tuned on my training set using cross entropy loss the input paragraphs, with the target paragraph being the input paragraph. The input paragraphs from the modified Brown Corpus described above were tokenized using the associated BERT cased tokenizer.

To identify which genre a given sentence was, I trained a "style classifier" on the latent representations to identify which attributes the sentence has. This classifier consisted of 3 linear layers, with leaky ReLU in between, as seen in Fig. 2. The classifier was trained at the same time as the auto-encoder, with the hopes that this would lead the auto-encoder to create latent representations that reflected the input genre more strongly. The cross entropy loss between the predicted label and actual label was used to optimize this classifier, as well as the "bag of words" similarity loss, as described in Eq. 1 below.

Once the auto-encoder and style classifier were trained, the style transfer itself was achieved by the iterative gradient descent method (Wang et al., 2019), (Liu et al., 2019). First, the style loss was calculated using the cross entropy loss of the trained style classifier's predictions with respect to the desired class. Then, a content preserving loss was calculated by penalizing any mismatch in the bag of words representation of each sentence. The

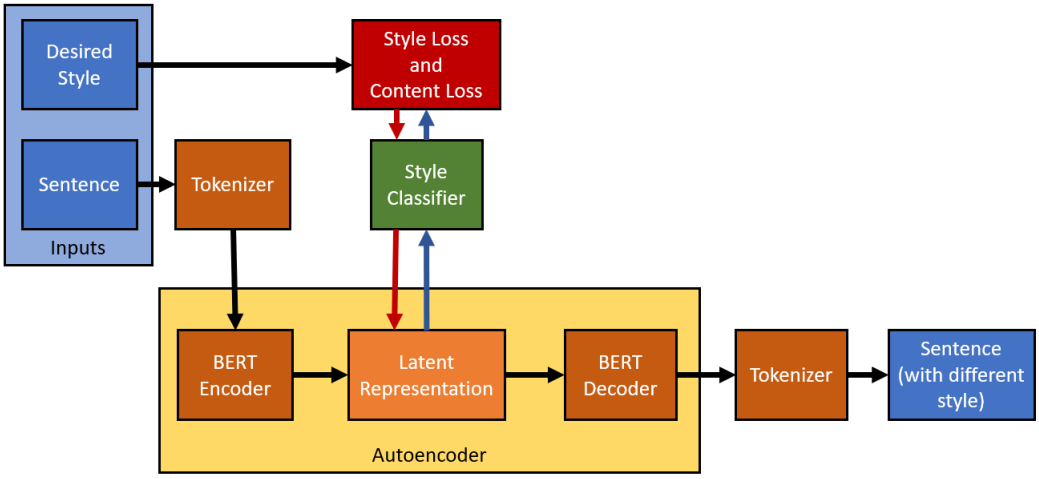

Figure 1: The high level model structure.

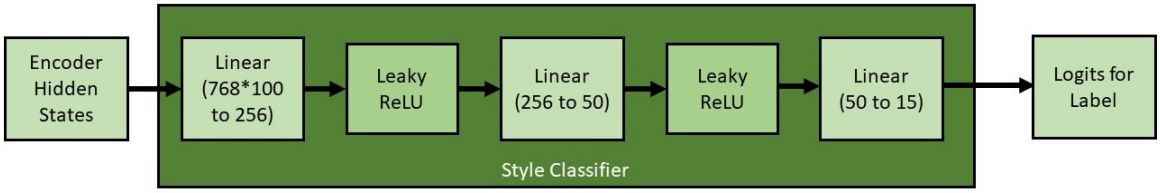

Figure 2: The style classifier structure.

bag of words overlap was computed with a simple iterative method, that could probably be optimized in the future. The actual loss function used to penalize the lack of this overlap was as shown in Eq. 1, where L_{BOW} is the "bag of words" loss, w_0 is the bag of words vector for the initial sentence, and w_n is the bag of words vector for the sentence represented by the current iteration of the algorithm. This is essentially finding the normalized average deviation from original sentence's bag of words representation.

$$L_{BOW} = mean\left(\frac{w_0 - w_n}{|w_0 - w_n|}\right) \quad (1)$$

The overall loss used in this calculation was as seen in Eq. 2, where L_{CLS} is the cross entropy loss of the style classifier as described above. The scaling factors λ_{CLS} and λ_{BOW} were used to regulate the ratio of these two types of loss, to fine tune the balance between style transfer and content preservation.

$$L = \lambda_{CLS} L_{CLS} + \lambda_{BOW} L_{BOW} \quad (2)$$

By using a backward pass on this loss function, gradient with respect to the latent representation (z) of the sentence could be calculated, and used to modify the latent representation as shown in Eq. 3, which was based on the method used in Wang et al, 2019 (Wang et al., 2019). The factor of ϵ was essentially a learning rate that could be tuned to increase or decrease the level of transfer.

$$z_i = z_{i-1} - \epsilon * \nabla z_{i-1} \quad (3)$$

I used this classifier to calculate gradients of the desired attributes on the latent representation, iteratively moving to a latent representation with the desired attributes using gradient descent. After 5 iterations, I cut the transfer off, to avoid spending too much time in this computation.

The model as described above was programmed using pytorch, and the autoencoder and classifier were trained using the hyper-parameters shown in the table below. To facilitate fine tuning a such a large model, I made use of the GreatLakes compute cluster to gain access to Graphics Cards that I wouldn't otherwise have access to.

Parameter	Value
Batch Size	16
Number of Epochs	4
Autoencoder Learning Rate	$1e-5$
Autoencoder Optimizer	Adam
Style Classifier Learning Rate	$1e-4$
Style Classifier Optimizer	Adam
Style Classifier Hidden sizes	256 and 50
BOW Loss Factor (λ_{BOW})	0.75
CLS Loss Factor (λ_{CLS})	0.35
"Style Transfer Learning Rate" (ϵ)	1.0

It took me a while to settle on this exact architecture, with the loss functions described above, and with joint training instead of training the classifier separately. Some of the other approaches involved trying to use a GPT-2 base, a BERT base, and BERT encoder and GPT-2 decoder, before I finally arrived on the BERT to BERT model described above. The python program used to train the model, as well as the dataset can be found in this GitHub Repository.

4 Evaluation

In the style transfer task, there are three main elements to assess: fluency of output, quality of style transfer, and semantic preservation of the input. Balancing the three of these can be very difficult, even if the model can do one or even two of them well (Jin et al., 2020).

To assess the output fluency, I calculated the perplexity of the outputs compared to the test set, using a pretrained GPT2 language model and the HuggingFace "evaluate" perplexity metric. This was the easiest way to get an estimate for the perplexity, but it really is estimating how well my outputs map onto the GPT2 language model, trained on a large dataset, rather than a language model trained just on my own dataset. I just ran out of time to train a seperate language model to get a more accurate perplexity score.

To assess the quality of the style transfer, I had planned to train a separate style classifier on the training set. The percentage of transferred paragraphs that fooled the classifier would be considered the accuracy for this metric. Of course, with something as subjective as genre, this is far from an objective standard for quality of style transfer, but it does make it possible to do without hiring actual people to assess the transfer quality. However, when I got to the task of actually training such a classifier, the performance was very low, about 10% accuracy. I think if I had tried to train an RNN-based classifier, instead of just a linear classifier, this would have had a better chance of working. I of course still can calculate the accuracy based on the latent space style classifier that is part of the model.

Without a set of gold standard labels, the BLEU score cannot be calculated to compare with previous work. Instead, to assess the content preservation, I evaluated the unigram word overlap, using Eq. 4, as in Liu et al, where w_x is the vector of word ids in the input paragraph, and $w_{\hat{x}}$ is the vector of word ids in the style transfered paragraph(Liu et al., 2019).

$$\textit{Unigram Word Overlap} = \frac{Count\left(w_x \cap w_{\hat{x}}\right)}{Count\left(w_x \cup w_{\hat{x}}\right)} \tag{4}$$

The above metrics were calculated for the best trained model, and the results are shown in the table below, where "Acc" refers to the style attribute classification accuracy, "PPL" refers to the perplexity, and "Overlap" refers to the unigram word overlap. A higher accuracy, lower perplexity, and higher overlap are desirable. For Liu et al, the reported metrics are from their style and content balance configuration on the Yelp sentiment analysis dataset. For Wang et al, the results are also for their model trained on the Yelp dataest. These will not be a perfect comparison, due to the different dataset, but because my model architecture is based on theirs it will be helpful to compare.

Model	Acc	PPL	Overlap
Mine	2.62%	585.2	33.57
Liu et al, 2019	92.3%	18.3	38.9
Wang et al, 2019	95.4%	46.2	N/A

Beyond the numerical results, a few of the outputs can be viewed to get a better picture of what is going on. For some example outputs, see Appendix A.

5 Discussion

As can be seen in the table of results, the performance of my model is much worse than that of the two models it is based on. The incredibly low accuracy is particularly noticeable, although partially explained by the fact that the other models it is being compared to had only a binary classification problem (positive or negative sentiment), rather than 15 classes. Also this metric is also just an approximation of the genre transfer accuracy, which

is calculated based on a classifier that I trained on a task that is highly subjective. As described above, the accuracy metric in general is difficult to assess without being validated by human scoring, which I did not have access to for this project.

The high perplexity and low word overlap are also notable, and are reflective of the poor performance of the autoencoding model itself. In general, the main contributor to the poor overall performance is mostly due to the poor auto encoding quality, which in turn is largely due to it being trained for only 4 epochs. I did train an autoencoder for 24 epochs. This model's latent space genre classifier achieved a 36.5% accuracy on predicting the genre, which is far better than randomly guessing. It also wrote sentences that were close matches for the input. For an example of these outputs, see Appendix A. Sadly, I didn't set a long enough time limit in the slurm job, and I forgot to save at checkpoints, so all the progress was lost. This was the last long job I was able to run before the GreatLakes servers either went down or were just completely inundated, so I was forced to do my validation on the model that was trained for only 4 epochs. This model had noticeably worse sentence outupt, though its classifier was 35.7% accurate. I think the reason that the genre classifier had similar performance between the models is that the encoder had fewer newly initialized weights to train than the decoder did, so it was already very good at extracting semantic information from the paragraph that would be useful for the classifier. The decoder had far more parameters to train, and so required far more training time. Without a solid decoder, any alteration of the latent paragraph representation just created even more gibberish.

Additionally, some of this poor performance certainly has to do with the dataset I tried to use– perhaps genre is too abstract a concept to use to achieve style transfer. With more time, I would be interested to try grouping the texts in the Brown Corpus into larger groups, perhaps news, nonfiction, and fiction, to see if with less granularity performance could be improved.

Beyond training the autoencoder and the latent space genre classifier for longer, I could also have likely improved the performance by experimenting more with different batch sizes and learning rates, as well as more complicated loss functions. In the actual iterative gradient descent modification of the latent paragraph representation, I could have varied the relative weighting of the two aspects of loss, as described in Eq. 2. This could potentially allow me to fine tune the content preservation and style transfer performance. Additionally, I could have used only the nouns when calculating the bag of words loss, which would allow other words to change more freely, since most semantic information that needs to be unchanged is likely to reside in the nouns (Liu et al., 2019). I could also have trained a second classifier to compute the bag of words loss, rather than doing a forward pass through the model to get the bag of words paragraph representation.

6 Conclusion

Overall, I think there is still a lot of promise in using this form of latent space vector modification to achieve style transfer. However, such large models take a vast amount of time and resources to train, and as such future investigations of more efficient methods can only help with the development of neural language models to be applied to tasks such as style transfer. Additionally, the style transfer task itself is already difficult to evaluate, due to the subjective nature of "style". For genre based style transfer in particular, it is difficult to define what exactly a genre is, so it is hard to fully evaluate the model's success, without using other language models to evaluate it. As such, an approach that relies on training a classifier to distinguish between the desired style attributes may always have some difficulty in the genre transfer task. A less granular grouping of the genres represented in the dataset, and the performance of such methods on other datasets would be interesting to see in the future. Despite these challenges, from what I have read and what I have seen the beginnings of through the course of this project, I am confident that style transfer models will continue to rapidly improve in the coming years.

References

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

Di Jin, Zhijing Jin, Zhiting Hu, Olga Vechtomova, and

Rada Mihalcea. 2020. Deep learning for text style transfer: A survey.

Dayiheng Liu, Jie Fu, Yidan Zhang, Chris Pal, and Jiancheng Lv. 2019. Revision in continuous space: Unsupervised text style transfer without adversarial learning.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Ke Wang, Hang Hua, and Xiaojun Wan. 2019. Controllable unsupervised text attribute transfer via editing entangled latent representation. In *NeurIPS*.

Yunli Wang, Yu Wu, Lili Mou, Zhoujun Li, and Wen-Han Chao. 2020. Formality style transfer with shared latent space. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2236–2249, Barcelona, Spain (Online). International Committee on Computational Linguistics.

A Appendix: Example Outputs

Some example outputs from the saved model, trained for 4 epochs, can be seen in Fig. 3. Notably, with an under-trained autoencoder, the output even before the style transfer is applied is pretty much gibberish. It just gets less intelligable after messing with the latent paragraph representation. In Fig. 4, the output from the training of the autoencoder that was trained for 24 epochs can be seen to be much closer to the original input sentence. This is why you should always save checkpoints–lesson learned.

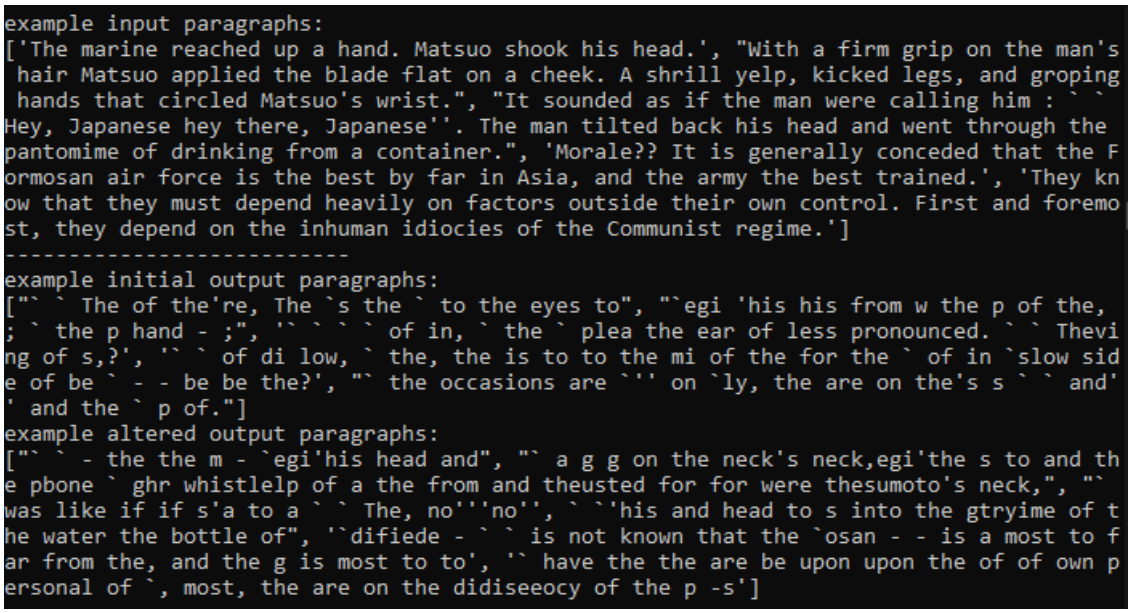

Figure 3: Example outputs from the trained model after the autoencoder was trained for 4 epochs. The section labeled "example input sentences" shows the paragraph the model is being applied to. The section called "example initial output sentences" shows the autoencoder output without style transfer, and the section called "example altered output sentences" is the output after the style transfer was attempted.

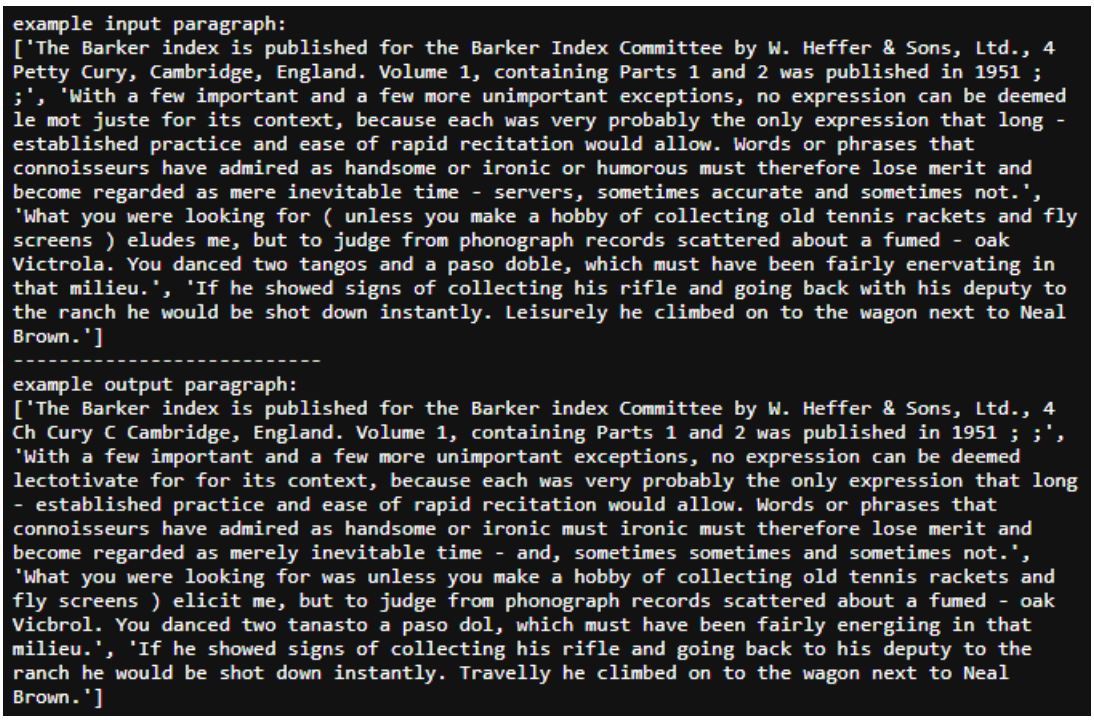

Figure 4: Example outputs from the autoencoder after 24 epochs. The section labeled "example input paragraphs" shows the paragraph the model is being applied to. The section called "example output paragraphs" shows the autoencoder output without style transfer. The evaluation of the actual style transfer was not done during this run, as I was expecting to save the model and run evaluation later.