BlindStyle: Unseen Style Transfer Structured as Paraphrase Generation

Milo Cason-Snow mcasonsnow@umass.edu
Bill Ray wray@umass.edu
Bowen Lei blei@umass.edu
Zhanna Kaufman zhannakaufma@umass.edu

1 Problem statement

Style is an integral component of a sentence indicated by a person's choice of words. Different people have different ways of expressing themselves, and they adjust their speaking and writing style to a social context, an audience, an interlocutor, or an occasion's formality. Text style transfer is defined as adapting or changing a sentence's style while preserving the original sentence's meaning.

Natural language generation (NLG) systems can allow for control over various attributes of generated text – for example, the sentiment/meaning of a sentence, and the style in which it is written. Recent neural text generation models in machine translation, image captioning, and dialogue have relied on massive parallel data. However, only non-parallel data – including collections of sentences from each domain without explicit correspondence – is available for many other domains. Many text style transfer problems fall into this category. The goal for these tasks is to transfer a sentence with one attribute to a sentence with another attribute, but with the same style-independent content, trained using non-parallel data.(Yang et al., 2018)

The problem we attempted to solve was to train a style-transfer model that could both be trained on a non-parallel dataset and operate on previously unseen styles (one-shot during inference).

2 What you proposed vs. what you accomplished

In our original idea, we proposed to model the text style transfer task as a controlled paraphrase generation problem (Krishna et al., 2020). In this problem, between the original style and new style of text, we would generate a neutral paraphrase from the sentence from which we wanted the content and a style vector from the the sentence from which we wanted the style. These pieces of information would then be combined and put through a decoder, which would be trained by reproducing the original sentence in the same style.

We ended up having some difficulty with the style-vector based approach, so we pivoted to a different approach using style prototype sentences which was trained on essentially the same goal, which we will discuss further in our approach section later in this paper.

The following is a list of what we were able to accomplish.

- Find and balance dataset for our use case
- Train BERT-based style-classifier on balanced dataset for style embedding extraction
- Pre-process data and prepare dataset by extracting style embeddings
- Train GPT2 based decoder with style-vector inputs (this did not end up working well)
- Train GPT2 based decoder with style prototype sentence inputs
- Perform both qualitative and quantitative error analysis

3 Related work

Text Style Transfer Text style transfer is a dynamic and popular field of research that has been further propelled by achievements in deep learning (Jin et al., 2021). Many transfer types have been researched, including genre, formality, complexity, and sentiment. Approaches include traditional linguistic approaches as well as more modern neural network approaches, and popular approaches differ based on the types of data available (Jin et al., 2021). Most datasets are either parallel,

where pieces of text and their direct translation are available, or non-parallel pieces of text in a singular style.

With parallel datasets, Seq2Seq models with encoder-decoder architectures have been used, implemented via LSTM, or transformer (Jin et al., 2021). Examples include implementation with a pointer network for English to Shakespearean (Jhamtani et al., 2017), and an expert filtered health sentence dataset for medical text simplification (van den Bercken et al., 2019).

Unfortunately, parallel datasets can be very difficult to find or create, especially for multiple styles at once, so mono-style datasets must be used here. Three of the most popular approaches to training on non-parallel data are disentanglement, prototype editing, and pseudo-parallel dataset construction (Jin et al., 2021).

Disentanglement and Prototype Editing Disentanglement generally involves encoding source text with some attribute into a latent representation, manipulating the representation to remove said attribute, and then decoding using the disentangled representation and the desired attributes (Jin et al., 2021). This often involves adversarial training (Sudhakar et al., 2019). However, it has past work has found that successful disentanglement, where the attribute can no longer be recovered from the representation, may not be achieved in practice (Lample et al., 2019). It may also result in content loss, while models improving content preservation suffer with implicit sentiment and have limited applications (Luo et al., 2019). For this reason, some work has moved away from this approach (Smith et al., 2019). Prototype editing involves replacing markers of some attribute with markers of the target attribute, then editing the final sentence to ensure fluency (Jin et al., 2021). One such approach, the Delete, Retrieve, Generate approach, utilizes the fact that style transfer can be accomplished by changing only a few markers (Li et al., 2018).

Pseudo-Parallel Data-Sets When no parallel dataset exists, certain methods of auto-generation can lead to effective training data(Jin et al., 2021).

One way to create pseudo-parallel data is through extracting semantically similar sentences from corpora of different styles (Jin et al., 2021), but a more popular way to do this is via corpus generation. One of the predominant methods of generation is iterative back-translation. Back-translation involves translating target text to source data (Sennrich et al., 2016) - some source x with attribute set y is encoded into z, and then decoded using a second set of attribute values ˜y into ˜x(Lample et al., 2019). ˜x is then encoded using y into x, learning the mapping of ˜x to x using attributes y. Originally developed for language translation, BT has also been used for style attributes (Prabhumoye et al., 2018; Lample et al., 2019). Some work has used back-translation as a warm-up to reinforcement learning, where forward and backward translation models train each other (Luo et al., 2019).

Our approach is largely based on Krishna et al., a group which approached the problem through controlled paraphrase generation (Krishna et al., 2020). The cited paper introduces the Style Transfer via Paraphrasing (STRAP) method. This method works in three stages. First sentences of different styles are fed through a diverse paraphrase model to create pseudo-parallel data. Then a style-specific inverse paraphrase model is trained to convert the sentences back into their original style, allowing them to learn stylistic features through reconstruction. Finally, the inverse paraphraser is used to perform the style transfer (Krishna et al., 2020). We use their structure as a foundation, but rather than an inverse paraphrase model which learns stylistic features, we attempt style agnostic paraphrasing. Instead, a separate model encodes the style and a decoder uses both the style encoding and the paraphrase to convert the input to the intended style. This approach of separating content from style is further based on a second group (Smith et al., 2019) which approached the problem of unseen text style transfer.

Unseen Text Style Transfer While much TST work has focused on a discrete set of possible styles (e.g., formal/casual, positive/negative), there has also been work done in inferring and transferring text to styles unseen during training. This involves matching style attributes unseen in training during text generation (Smith et al., 2019). Our implementation is part based on the work of Smith et al. in this area (Smith et al., 2019). In their work, they decoupled the classifier from the style transfer architecture. So the classifier only produced a distributed representation of the target attribute, and existing pre-trained supervised representations could be reused. To create the target

representation, they use a back-translation objective. Our work is similar in that we are decoupling the style representation from the target representation, but we use a classifier for the style representation and a paraphraser for the target representation.

4 Your dataset

4.1 Training Dataset

The dataset which we used for training all of our models was the Corpus of Diverse Styles (CDS) developed by (Krishna et al., 2020) for their style transfer through paraphrase generation work. This corpus includes AAE tweets, English tweets, the King James version of the Bible, the Corpus of Historical American English for the 1810s, 1820s, 1890s, 1900s, 1990s, and 2000s, James Joyce works, lyrics, romantic poetry, Shakespeare works, and lines from Switchboard, a corpus of conversational speech and text curated by (Godfrey et al., 1992). This dataset is provided by Krishna et al. on an openly accessible google drive[1].

The CDS contains $14,018,731$ training, $393,727$ validation, and $14,412,458$ sentences total from each of the above 11 styles.

4.2 Evaluation Dataset

For evaluation, we used both parallel and non-parallel datasets. This was mainly because parallel datasets are difficult to find and difficult to make. We were able to find two parallel datasets: the first was a collection of bible versions curated by Carlson et al.[2] (Carlson et al., 2017). This contains 8 bible versions in total: The American Standard Version, the Bible in Basic English, The Darby bible, the Douay–Rheims American Edition, the King James Version, the Lexham English Bible, the World English Bible, and Young's English Translation. The group used this corpus for their style transfer work because it is highly parallel, as sentences are aligned, and the versions are stylistically distinct (Carlson et al., 2017). A concern for this dataset would be that the CDS dataset which we used for training contains bible sentences; however, these sentences are all from the King James Bible (Krishna et al., 2020), and as is posited by Carlson et al., the different versions are written with different intentions and in different periods of time (Carlson et al., 2017). This makes them different enough so as to count as "unseen" styles.

When evaluating, we chose two sets of bibles which were particularly different from each other stylistically: ASV and BBE, and YLT and DRA. The YLT Bible was published in 1862, and was translated from Hebrew and Greek - it is meant to be a literal translation, and is very formal and sometimes Shakespearean in its language. The DRA version is translated from the Latin Vulgate and reads much more straightforwardly. Similarly, the ASV is more complex and formal than the BBE, as the BBE is written in basic English, and meant to be simple to understand, using only 850 different words.

Another parallel dataset we were able to find was "The Knight's Tale", from the Canterbury tales by Geoffrey Chaucer, which we were able to obtain from Harvard's Geoffrey Chaucer website[3]. This was written in middle English in the late 1300s, and so would be very unlikely to be contained in any of the above mentioned sources for the CDS dataset. It also makes for a particularly interesting dataset, since the original was written in Middle English. This means that there are many words used which are not used in English today, making it a completely unseen "style" for the models. However, the language is not so different that we would call this a language translation problem.

In order to perform evaluation on parallel datasets, we used sentences from one style as the content input and sentences from another style as the style input. The model was then evaluated on how well it was able to produce the parallel equivalent of the content input.

The non-parallel dataset which we used was the CDS test set created by Krishna et al. and preprocessed by us. This test set does not contain unseen styles, but does give us an idea of how well our model is able to translate to the styles that it has been trained on with only the style token prepended. When evaluating on this dataset, we used some sentence in the set as input to the paraphraser, and then made sure to use a different sentence of the same style as input to the encoder. Because of the wide variety of styles in the CDS dataset, and the similarity of many styles that we see in natural language (i.e, 1990s vs 2000s), this can help us understand how effective our model

[1] https://tinyurl.com/yhr9c938

[2] https://tinyurl.com/5n7rh3na

[3] https://chaucer.fas.harvard.edu/pages/knights-tale-0

might be in general.

In addition to the test set, we prepared three non-parallel datasets obtained from project Gutenberg: "The Aeneid"[4], "Alice in Wonderland"[5], and Moby Dick[6] - while we were not able to use these datasets due to time constraints, we feel that they would be useful additions to a set of analyses on this approach.

4.3 Data preprocessing

For the training CDS dataset, much of the preprocessing was done by the original team (Krishna et al., 2020). The dataset was sorted into the 11 relevant styles used for training, meaning that we could use the category they were sorted into (in other words, where the data was sourced from) as a style label, and we did not have to perform any annotation. When looking at the data, we noticed that there were far more instances of some styles than others. We were concerned that this would lead to bias in the classifier, so we balanced the dataset by finding the style with the smallest number of corresponding elements, and limiting the number of sentences for all other styles to this value. This resulted in about 24,000 sentences per style, and around 270,000 sentences total that were used for fine-tuning. We did this for the training, validation and test sets. We then randomized the ordering for each set so that during training time, the model would encounter a variety of different styles in each batch. We also paired each sentence with another sentence from the same style which could be used to extract a style vector from that style - a design choice we discuss further in our approach section below.

4.4 Evaluation set preprocessing

The evaluation sets we used were pre-processed in essentially the same way as the training set, except that instead of pairing each sentence with a sentence from the same style, each sentence was paired with a sentence from the style to which we were attempting to transfer it.

4.5 Qualitative dataset analysis

As with any dataset, we noticed some qualitative properties of this dataset that may have had an effect on our final results. One of the most important aspects of the dataset that we noticed initially was that the difference between two sentences in two different styles could be much more subtle than what you might encounter in another seq2seq problem such as machine translation. Between different styles, the same words and phrases are often used, meaning that these styles are difficult to distinguish. This was especially true because some of the different styles which we were working with in our dataset are very similar to each other, and sometimes the distinction between them is nearly arbitrary. For example, two of the styles in the CDS dataset are text from the 1990s and the 2000s. Some examples in the dataset are things like:

- 1990s: "How are you?"
- 2000s: "How are you doing?"

It is difficult to make a case that these two sentences are easily identifiable as from different styles; the distinction between them seems arbitrary. Additionally, with a relatively short sentence such as the one seen above, there is not a significant amount of information present about the underlying style.

Other difficulties arise in sentences that rightfully belong to their style, but could just as easily belong to another style. For example, consider this sentence from the dataset which is labeled as an example of AAE (African-American English):

- "Fumble bitch!!!! #Eagles!!!"

This sentence has a hashtag in it, suggesting that it was likely sourced from a social media site, and could therefore be easily confused with the "Twitter" style category.

Despite these difficulties, however, there are enough differences between the different styles and enough diversity within the styles that we feel that this was still an effective dataset to train on.

4.6 Discussion of the separability of style and content

Separating the style and content of a piece of text into distinct representations can be a difficult task. This is sometimes accomplished by trying to perform an adversarial disentanglement task, as discussed in the "Related Work" section above.

Krishna et. al found that their paraphrase model was effective at generating neutral paraphrases which were difficult to distinguish (Krishna et al.,

[4]https://tinyurl.com/f73rzbjr
[5]https://tinyurl.com/42ym9ym5
[6]https://tinyurl.com/42ym9ym5

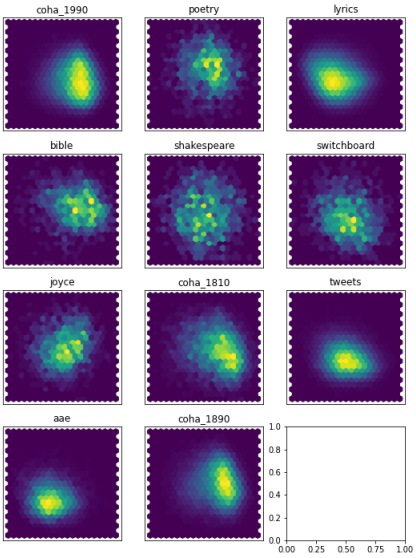

Figure 1: Hexbin plot of PCA of sBERT output from paraphrased sentences in dataset.

2020). We initially took an approach to our project where we attempted to extract a sentence-level content embedding using sBERT (Reimers and Gurevych, 2019). Although we ultimately decided not to use these embeddings as inputs to our decoder, we used sBERT to analyze the separability of style from content. We found that even after put through the paraphrase model and then through sBERT (to get an embedding of a sentence), different styles of sentences were still somewhat distinguishable. We have included a figure of a hexbin plot of PCA done on the outputs of this sBERT on a portion of the data. The figure shows that there are still some significant differences between the different classes that are included in the dataset.

5 Baselines

As discussed in our initial proposal, the baseline(s) that we chose for our model were very simple but effective, not requiring any kind of complicated model building/training. Both baselines that we analyzed were copying-based baselines, which Krishna et al. found to be effective baselines in measuring the real performance of style transfer models (Krishna et al., 2020). The first of these baselines was copying the content input, and the second of these baselines was copying the style input (input in this case meaning the sentence used to generate either the style or content representation). These baselines are very effective up to a certain point because each one perfectly accomplishes one of the main goals of the model, of which there are two: preserving the semantic content of the sentence, and outputting a sentence in the correct style. For styles that are similar to each other, copying of the content input performs very well, since many of the words and phrases used may be very similar or the same.

6 Your approach

Our approach consists of three models: a GPT2 Paraphraser, a style encoder realized via a BERT Classifier, and a GPT2 Decoder. Figure 1 shows the training and evaluation steps of the system. A sentence is input into the paraphraser, which outputs a set of paraphrase tokens. A second sentence in the desired style is input to the style encoder - this encoder is realized via a classifier, and the final hidden state, corresponding to the CLS token, of the classifier is taken as the style token. The style token is then prepended to the paraphrase tokens, and the full token array is input into the decoder. This decoder should then output the content of the paraphrased sentence in the encoded style.

During training, as well as evaluation on non-parallel corpora, the input to the style encoder is in the same style as the input to the paraphraser. However, in order to improve the robustness of the model and to ensure that no content information could make its way into our style representation, we chose to use a *different* sentence from the content sentence, but one that was in the same style. In this case, the objective for the decoder is to output the source sentence of the paraphrase, in its original style. During evaluation on parallel corpora, as well as intended final operation, the input to the style encoder is in a different and unseen style. In this case, the objective of the decoder is to output a sentence with the content of the paraphrased sentence, in the unseen style.

6.1 Style Encoder

The style encoder was implemented by finetuning BertForSequenceClassification model on huggingface[7]. The inputs were sentences from the pre-processed CDS dataset, and the objective was to classify these sentences into one of the 11 given styles. The model was trained using AdamW from

[7]https://tinyurl.com/zhr77txn

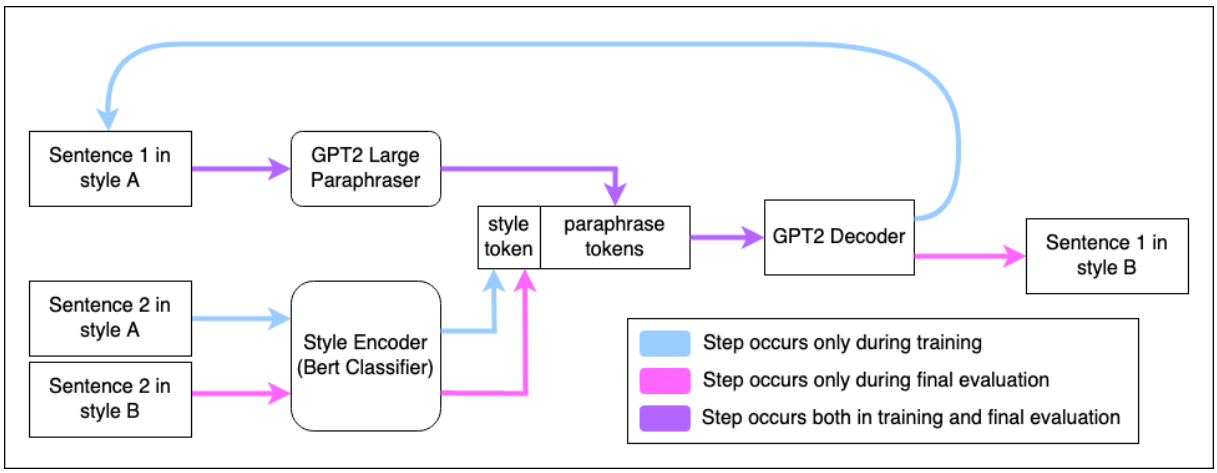

Figure 2: This diagram shows the encoder/decoder process from input to output during training and for final output (used for evaluation). During both training and evaluation, some sentence 1 in some style A is brought through the GPT2 large paraphraser, which will output a string of paraphrase tokens. A style token is prepended to this string which is obtained from the final layer of a BERT classifier which acts as a style encoder. During training or evaluation with non-parallel corpora, this style token is obtained using a second sentence in style A, and during evaluation with parallel corpora the sentence is in a different (unseen) style. The style and paraphrase tokens are then brought through the GPT2 decoder. During training and non-parallel evaluation, the objective is for this decoder to output the original sentence 1 in style A. During evaluation on parallel corpora, the objective is to output sentence 1 in the unseen style, style B.

the transformers library as an optimizer, with default values aside from a learning rate of 2e-5 and an epsilon of 1e-6. The batch size was 32. These parameters were obtained by varying parameters as recommended in the seminal paper on BERT (Devlin et al., 2019), as well as work done on fine-tuning BERT (Mosbach et al., 2020). We found that changing hyperparameters had little effect on the accuracy of the fine-tuned model, with training and validation performance ranging between .82 and .84 using a flat accuracy calculation. Varying the number of epochs also had little effect on performance, so only 1 epoch was used. The final model had a training and validation accuracy of .84. After fine-tuning, the model was frozen. When training the decoder, the the frozen model was loaded and the final hidden state after each input was used as the style encoding.

6.2 Paraphraser Encoder

In order to limit the number of models that we were actively training, we decided to use the paraphrase model trained by Krishna et al for their paraphrase-based style transfer model. This model was fine-tuned from GPT2 large in order to produce an output sentence in a neutral, paraphrased style. We loaded their model and ran it in Colab/Jupyter notebook in order to generate paraphrases for training and testing.

6.3 Initial Decoder

The bulk of the work of our project was spent on designing and training the decoder that we used, whose goal it was to take a style representation vector and a content sentence in the form of a paraphrase and output a sentence in that style.

This decoder is intended to act as a kind of style-agnostic inverse paraphrase model, where it performs the opposite of the initial paraphrase model developed by Krishna et al by taking in a style vector.

In order to create our decoder, we decided to use a decoder-only approach for seq2seq text generation. Although many seq2seq models use an encoder-decoder architecture, we were influenced by Krishna et al (Krishna et al., 2020) in choosing a decoder-only architecture, which allowed us to create a seq2seq inverse paraphrase model by fine-tuning the GPT2 language model, specifically GPT2-large (Radford et al., 2019), which we downloaded from huggingface using the transformers library [8]. In order to train an encoder-free seq2seq model, we employ a method used by Krishna et al which is outlined in a previous paper. This method involves concatenating the input sequence (in this case, the paraphrase of the original content input sentence) with the output sequence,

[8] https://tinyurl.com/y5ae3x2t

with a special divider token in between. This divider token is a special beginning-of-sequence token for the generated text. Then, the embeddings are prepared in a manner consistent with the original GPT2 approach in which the token embeddings are added to positional embeddings, generating the final input embeddings (Wolf et al., 2019).

In order to include the style vector in the input to the GPT2 decoder, we decided to include this style vector as a token embedding input to the decoder. This style vector was extracted by taking the final hidden state of the [CLS] vector of the BERT style classifier. Because this [CLS] vector was used for style classification, we reasoned that it would contain a large amount of information about the style that was encoded. The [CLS] hidden state is of size 768, so in order to make it the correct size to be input to the decoder model and to provide flexibility of input, we put a linear layer on top of it projecting to a size of 1280, which is the input embedding size for GPT2-large. This embedding was then included as if it were a normal token input embedding at the end of the token sequence, appended to the end of the list of normal token embeddings.

We expect this model to outperform the baselines that we have established in being able to passably preserve the content of text while transferring a new style to that content.

6.4 Decoder implementation and training details

The decoder was implemented by fine-tuning the GPT2-large present on huggingface via the transformers library, along with PyTorch. Tokenization was done using the GPT2 tokenizer from transformers. During the tokenization process, we padded each input and output sequence to be no longer than 50 tokens, and truncated any longer than that length. Specifically, the model that was fine-tuned was the huggingface GPT2LMHeadModel [9]. Because of the specific way that we wanted to use the input embeddings to the decoder model, we had to handle the embeddings manually and provide the input embeddings to the model instead of the input ids, which is normally standard. We trained the decoder on the next word prediction task using the cross-entropy loss, with a batch size of 20 and a learning rate of 5e-5, along with the AdamW optimizer and the learning rate warmup that is standard for transformer-based model training.

The model was developed and trained on a local RTX 3060 GPU, as well as a cloud-based A6000.

6.5 Decoder difficulties

Ultimately, we were not successful in training the decoder with this method. We had some difficulty with the training process, specifically with decoding coherent text. We suspect that there may have been some bug in our training algorithm; however, we were not able to figure out what this bug was. In particular, we had difficulty training the model on next word generation and decoding coherent outputs due to the specific way in which we input embeddings to the model. We ended up getting nonsense output results when decoding from this model - some example results were:

- ”ENTSENTSENTSENTSENT....”
- ”<bos> <bos> <bos> <bos>....”
- ”(((((((....”

6.6 Alternative Decoder Approach

Because of the lack of success with the initial decoder approach, we decided to pivot to a different and slightly simpler approach to training a decoder. This approach is a slightly modified version of the approach used by Krishna et al (Krishna et al., 2020) and initially described in Wolf et al (Wolf et al., 2019). We performed fine-tuning on GPT2 for decoding, but instead of inputting a style vector (since that was what was causing us some difficulty with our first model), we decided to instead choose a ”prototype sentence” for a specific style to serve as an example of that style and provide that as input to the decoder, along with the neutral paraphrase. This style prototype was sampled as a different sentence from the same style as the original input for training purposes. Specifically, we fine-tuned GPT2-small on the next word prediction task on a sequence that look like this:

- ”[style prototype] [separator] [paraphrase] [separator] [original input text]

A separator token was included in between the style prototype and the paraphrase and between the original input text (the ground truth) in order to distinguish the sections. The attention mask was

[9] https://tinyurl.com/yxvt3xa2

set so that the model could train on predicting the original input text.

This model was trained using the transformers library, specifically the GPT2LMHeadModel and the GPT2Tokenizer, along with utilities such as the Trainer class. It was fine-tuned on 10000 examples using a learning rate of 5e-5 and the standard warm-up learning rate, as well as the AdamW optimizer (using the transformers Trainer API).

After decoding, the output sentence is then taken as everything output after the last separator token.

6.7 Results

Despite the somewhat disappointing results we achieved with our initial approach, we did achieve some promising results with our second approach. The model was able to learn that the most important aspect of style transfer is preserving the content, as by examining the results it seems to show that the model often outputs something just slightly different than the input paraphrase, with slight style augmentations to nudge it in the direction of the input style. One downside of the model that we generated is that it seems that the input style does not have quite as much influence as we would like, so a future approach of this work might be to try to enhance the influence of the input style somehow (maybe by getting something like our initial approach to work).

The main limitations of our baseline copying-based models is that they do not accomplish the task of actually transferring the style (the most effective baseline model is the one that copies the input sentence, as this preserves the semantic content but does not transfer the style at all). The prototype style model suffers from a somewhat similar problem in that the outputs that it gives are often in a neutral style, though it does show promising signs of some style transfer, which we discuss more in the error analysis section below.

We think that some of the limitations of our approach came down to the limited amount of information that a single sentence or phrase can provide about an unseen style - it can often be difficult to tell what style is present even from a human perspective. Perhaps a future approach could weight the training somehow, or include more information about the style in the input somehow. We believe that our findings support the initial choice of Krishna et al (Krishna et al., 2020) to use style-specific inverse paraphrase models, as they found this to be more effective.

7 Error analysis

We performed two specific types of error analysis: running the model on common metrics that are used to measure output performance of generative language models (quantitative error analysis), and qualitative error analysis by looking at the data.

For our quantitative error analysis, we included two different metrics, similar to what was included in Krishna et al, in order to measure different aspects of the abilities of our model. We used our style classifier to measure BLEU and ROUGE to measure the both the model's reconstruction ability on the test portion of our non-parallel dataset and also to measure the content preservation ability on parallel datasets.

Dataset	Rouge-2	BLEU
YLT->DRA Base	0.220	0.113
YLT->DRA Dec	0.124	0.027
AST->BBE Base	0.288	0.169
AST->BBE Dec	0.117	0.036
KN Mod->Orig Base	0.429	0.113
KN Mod->Orig Dec	0.002	0.109

In table 1, an arrow means the first dataset was decoded into the style of the second. YLT, DRA, BBE, and AST are Bible versions. KN stands for "Knight's Tale", where Knight's Tale was decoded from the original style into the modern. "Base" here means baseline, or the comparison of the parallel sentences in two datasets without any decoding done. "Dec" indicates that the first dataset was decoded into the second.

The Mean Rouge score for our non-parallel training set was 0.200, and the Mean BLEU score was 0.079.

7.1 Qualitative/Manual Results Analysis

When we looked at our results manually, we saw that our outputs were very close to the paraphrased sentences, but often with elements changed to reflect some of the style of the original sentence. This was not fully unexpected to us, as we began to see during evaluation that the style encoded token was not sufficient to fully transform a paraphrase back into a source sentence. However, we certainly saw some elements of style transferred. For instance, with the original sentence:
"On Thorfinn, however, all these negatives have

been ameliorated to the point that you hardly notice."
And the paraphrased sentence:
"however, all of these negative aspects were reduced to a point where you barely noticed."
Our output was:
"It was, however, that all these negative aspects were reduced to a point where you barely noticed."
Here, we can see that the output has more elements of formality than the paraphrased sentence, but is not quite as formal as the original input. Similarly, with the original sentence:
"Thou art fairer than the children of men: grace is poured into thy lips: therefore God hath blessed thee for ever."
And the paraphrased sentence:
"you're more beautiful than the children of men: God bless you."
Our model outputs:
"And thou shalt be more beautiful than the children of men: God bless thee."
Here it is even clearer, that there are some elements of the style transferred: specifically, we have "you" replaced by "thou" and "thee", "are" replaced by "shalt be", and capitalization added. In fact, in this case, it may be that one of the largest reasons for the difference between the output and input sentences is the content lost during paraphrasing.

Some of the most interesting results come from looking at the Knight's Tale dataset. The paraphraser was not very able to paraphrase middle English, though it was often surprisingly close. For instance, for a sentence like:
"Hym thoughte that his herte wolde breke,",
it output:
"but he was a little bit of a hiccup",
while the actual meaning was:
"It seemed to him that his heart would break".
However, for:
"Why cridestow? Who hath thee doon offence?",
the output was:
"who's the one who's offended you?",
and the true translation was:
"Why didst thou cry out? Who has done thee offence?"
- so the meaning was kept. Because we were translating to a "modern" style, the decoder often simplified the sentences even further ("I'm writing this list of my list of things to write." -> " I wrote this list of things to write"). However, it often devolved into repetition. We were more interested to see what our decoder would do when attempting to translate into the "style" of Middle English, with a paraphrase of the modern equivalent sentence. Unfortunately, most outputs were very close to the paraphrases. We did observe that the decoder replaced "you" with "thee" and "your" with "thy", so the greater formality of the output than the input had an effect, but it was not able to input any Middle English words. Nevertheless, there were cases where it was able to output something quite close to the reference output: with the input:
"This is thy mortal foe, this is Arcite,"
And the paraphrase:
"it's your mortal enemy, this is Arctite, Arctite, Arctite, Arctite, Arctite, Arct"
It was able to output:
" It is thy mortal foe, this is Arctite,"
Which was incredibly close to the actual reference output of:
"This is thy mortal foo, this is Arcite,"
notwithstanding the inability to produce a word like "foo".

8 Contributions of group members

List what each member of the group contributed to this project here. For example:

- Bill: did data collection and pre-processing for the training set, set up codebase, found and explored a semantic encoder and worked on GPT2 decoder
- Milo: Helped find and train the GPT2 decoder model, built prototype style decoder model, worked on writing paper
- Zhanna: did data collection for evaluation, trained BERT style encoder, worked on writing paper, ran evaluations for analysis
- Bowen: implemented majority of error analysis and evaluation approach
- all: came up with design as a team

9 Conclusion

There were several important technical ideas that we took away from this project. One of those ideas is that different aspects and attributes of of a piece of text are inherently entangled, and it can be difficult to separate them without distorting these attributes somewhat. Another thing that we learned

was that information about a particularly subtle attribute such as the style of a text can be difficult to extract, especially from a short piece of text.

We faced a number of difficulties while working on this project, probably the biggest of which was difficulty with the technical aspects of modifying a pretrained model to deal with a style-based embedding in the way that we wanted. However, this experience allowed us to see how text embeddings and large language models work at a lower level, so it was still a valuable experience despite the lack of success with that specific technique.

Finally, and perhaps most importantly, we learned the importance of leveraging existing work, building on top of existing effective models and techniques, and keeping things simple when possible, since that often seems to lead to better results.

References

Carlson, K., Riddell, A., and Rockmore, D. N. (2017). Zero-shot style transfer in text using recurrent neural networks. *CoRR*, abs/1711.04731.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding.

Godfrey, J., Holliman, E., and McDaniel, J. (1992). Switchboard: telephone speech corpus for research and development. In *[Proceedings] ICASSP-92: 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 517–520 vol.1.

Jhamtani, H., Gangal, V., Hovy, E., and Nyberg, E. (2017). Shakespearizing modern language using copy-enriched sequence-to-sequence models.

Jin, D., Jin, Z., Hu, Z., Vechtomova, O., and Mihalcea, R. (2021). Deep learning for text style transfer: A survey.

Krishna, K., Wieting, J., and Iyyer, M. (2020). Reformulating unsupervised style transfer as paraphrase generation.

Lample, G., Subramanian, S., Smith, E., Denoyer, L., Ranzato, M., and Boureau, Y.-L. (2019). Multiple-attribute text rewriting. In *International Conference on Learning Representations*.

Li, J., Jia, R., He, H., and Liang, P. (2018). Delete, retrieve, generate: a simple approach to sentiment and style transfer. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1865–1874, New Orleans, Louisiana. Association for Computational Linguistics.

Luo, F., Li, P., Zhou, J., Yang, P., Chang, B., Sui, Z., and Sun, X. (2019). A dual reinforcement learning framework for unsupervised text style transfer.

Mosbach, M., Andriushchenko, M., and Klakow, D. (2020). On the stability of fine-tuning BERT: misconceptions, explanations, and strong baselines. *CoRR*, abs/2006.04884.

Prabhumoye, S., Tsvetkov, Y., Salakhutdinov, R., and Black, A. W. (2018). Style transfer through back-translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 866–876, Melbourne, Australia. Association for Computational Linguistics.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners.

Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. *CoRR*, abs/1908.10084.

Sennrich, R., Haddow, B., and Birch, A. (2016). Improving neural machine translation models with monolingual data.

Smith, E. M., Gonzalez-Rico, D., Dinan, E., and Boureau, Y.-L. (2019). Zero-shot fine-grained style transfer: Leveraging distributed continuous style representations to transfer to unseen styles.

Sudhakar, A., Upadhyay, B., and Maheswaran, A. (2019). "transforming" delete, retrieve, generate approach for controlled text style transfer. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3269–3279, Hong Kong, China. Association for Computational Linguistics.

van den Bercken, L., Sips, R.-J., and Lofi, C. (2019). Evaluating neural text simplification in the medical domain. In *The World Wide Web Conference*, WWW '19, page 3286–3292, New York, NY, USA. Association for Computing Machinery.

Wolf, T., Sanh, V., Chaumond, J., and Delangue, C. (2019). Transfertransfo: A transfer learning approach for neural network based conversational agents. *CoRR*, abs/1901.08149.

Yang, Z., Hu, Z., Dyer, C., Xing, E. P., and Berg-Kirkpatrick, T. (2018). Unsupervised text style transfer using language models as discriminators. *Advances in Neural Information Processing Systems*, 31.