Style Transfer with Non-Parallel Corpora

Gus Liu **Pol Rosello** **Ellen Sebastian**
{gusliu, prosello, ellens2}@stanford.edu

Abstract

We investigate the problem of *style transfer*: Given a document D_1 in a style S_1, and a separate style S_2, can we produce a new document D_2 in style S_2 which preserves the meaning of D_1? We describe a novel style transfer approach that does not rely on parallel or pseudo-parallel corpora, making use of anchoring-based paraphrase extraction and recurrent neural language models. We analyze our results qualitatively and quantitatively. Our approach suffers when there is a lack of topical overlap between documents in both styles. We suggest ways in which our system could be improved.

1 Introduction & Problem Statement

Recently, the idea of style transfer has received considerable attention, especially in the field of computer vision. Gatys et al. (2015) use neural networks to combine the *content* of one image and the *style* of a second image to produce a new image. This makes it possible, for example, to transfer the painting style of Van Gogh onto a vacation picture with high perceptual quality. However, the techniques explored in their paper are not directly applicable to the domain of natural language processing.

A successful style transfer model (as applied to natural language processing) would take as inputs a document D_1 in a style S_1 and a separate style S_2, and would produce a new document D_2 in the style of S_2 which preserves the meaning of D_1. Style-transfer models would be useful in automating tasks such as author obfuscation, passage simplification, or publishing a document to multiple audiences. To our knowledge, past approaches to this task have used a supervised approach, making use of parallel or pseudo-parallel monolingual corpora. However, in most cases, parallel text between two styles does not exist. Therefore, in our work, we attempt to perform style transfer by training our model on non-parallel input corpora in both styles.

In this paper, we will detail our progress toward this goal, which involves training neural language models on corpora of various styles and implementing and adapting Pasca & Dienes (2005)'s anchor-based paraphrase extraction algorithm. Our approach is divided into two tasks:

1. Create a database of paraphrases $\{(p_{a1}, p_{b1}),..., (p_{an}, p_{bn})\}$, where p_{ai} is extracted from style S_1 and p_{bi} from S_2. We initially build a paraphrase-extraction algorithm based on the work of Pasca & Dienes, but ultimately use pre-extracted paraphrases from PPDB (2013) in order to increase the number of paraphrases available for Task 2.
2. Given a new text, replace some instances of phrases in $\{p_{a1}, ..., p_{an}\}$ with their corresponding phrase in $\{p_{b1}, ..., p_{b2}\}$ in the optimal way to preserve meaning and transfer style. We approach this task by choosing a series of replacements that maximize the paraphrased sentence's probability under a language model of S_2. This is a difficult and computationally-intensive problem because many replacements overlap with each other and the space of possible replacement combinations is very large.

2 Literature Review

Style transfer is a relatively unexplored space. To our knowledge, only one paper has attempted style transfer: Xu (2012) used parallel Shakespeare/modern English text to build a bidirectional style transfer system using typical machine translation algorithms. We note that in our work, since we do not use parallel text, we cannot use traditional machine translation algorithms that rely on the existence of a dataset of sentences in both styles (or languages). Traditional evaluation metrics such as BLEU are therefore also unavailable to us.

Although style transfer has not been studied extensively, the related task of paraphrase extraction has. All paraphrase extraction systems use either parallel text (e.g. multiple translations into English of a single non-English text) or pseudo-parallel text (e.g. news articles about the same event). For example, Barzilay & McKeown (2001)'s algorithm uses aligned sentences from parallel corpora, paraphrase pair seeds, contextual features, and scoring for positive and negative strength of those features. Barzilay & Lee (2003) generate paraphrases of novel input sentences using a "word lattice" algorithm. Instead of parallel data, they use "comparable" corpora, which is closer to our task. Pasca & Dienes (2005)'s work, which uses pseudo-parallel corpora and an anchoring system based on shared paraphrase surroundings, forms the basis of our paraphrase extraction system, which we discuss in depth in Section 4.

3 Data Collection

Our first task was to collect a number of stylistically-distinct parallel and non-parallel corpora on which to train and evaluate our algorithms.

Since Donald Trump's style was the original inspiration for our project, we collected a number of campaign speeches from WhatTheFolly.com (2016). The speeches were sanitized of annotations and words spoken by anyone other than Donald Trump. They total 422,653 words.

We downloaded 3 versions of the Bible from Christian Classics Ethereal Library (2015). These corpora are useful because they are parallel at the word and phrase level. They provide the opportunity to evaluate our algorithms both on completely parallel data and on non-parallel topically related data (by using non-overlapping subsets of each version as the source and destination styles).

Finally, we downloaded 12 Charles Dickens novels and 11 Mark Twain novels from Project Gutenberg (2016). These total 935,614 words for Twain and 2,710,985 words for Dickens. They provide the opportunity to evaluate our algorithm on very large non-parallel and unrelated corpora.

We also make use of an existing paraphrase database called PPDB (2013). We used the "small" version of PPDB 1.0, which contains the 6.8 million better-scoring, high-precision paraphrase rules in the full database's 169 million rules. The database contains lexical paraphrases (i.e. one word to one word), phrasal paraphrases (i.e. multi-word phrases), as well as syntactic paraphrases which contain nonterminals.

4 Methods

4.1 Anchoring algorithm for paraphrase extraction

We implemented a basic "anchoring" algorithm for paraphrase extraction, adapted from Pasca & Dienes (2005). It operates on the assumption that if $p_a \in S_1$ and $p_b \in S_2$ appear surrounded by the same words, they are likely paraphrases. A large number of n-grams are separated into 3 portions. The first and last portion are the "anchor text", and the middle portion is the "variable fragment", which is a potential paraphrase.

For example, if "Hillary Clinton defeated Bernie Sanders" appeared in S_1 and "Hillary Clinton Triumphed over Bernie Sanders" appeared in S_2, then we would mark (defeated, triumphed over) as paraphrases.

The unoptimized algorithm is summarized as follows:

1. Parse 2 large corpora into sentences, then further into tokens using NLTK tokenization. Prepend each sentence with a sentence-start tag "$S" and append with a sentence-end tag "$E".
2. Extract all possible n-grams of length $2L_c + min_p$ to $2L_c + max_p$ from each corpus, where no $n$gram can overlap separate sentences. Here, L_c is the length of the anchor, and 2 or 3 is used in our analyses. Min_p and max_p are the minimum and maximum paraphrase length, here 1 and 5.
3. Extract all possible (anchor, variable fragment) pairs from the sequence of n-grams. For each n-gram, the anchor is defined as the first and last L_c tokens, and the variable portion as the middle min_p to max_p tokens. Collect all pairs (v_1, v_2) of variable fragments appearing at least once with the same anchor fragment. v_1 is constrained to come from S_1 and v_2 from S_2.
4. Output the pairs (v_1, v_2) ranked by the number of times they appear with the same anchor text.

The results of this unoptimized algorithm appear in Table 1 together with the results from our optimized algorithm.

4.1.1 Optimizations on Paraphrase extraction

The major weakness of the unoptimized anchoring algorithm was insufficient overlap of anchor text in nonparallel, unrelated corpora. We therefore made several attempts to "genericize" the anchor texts (while leaving variable portions intact).

First, we note that unlike in Pasca & Dienes (2005)'s corpus of related news articles, the named entities, such as people and places, vary greatly between unrelated texts. While a corpus of Twain novels may talk about people named Huck and places like Mississippi, a corpus of Dickens novels makes no mention of theses specific named entities. We therefore used a named-entity tagger to replace named entities with their tags. Words that are not tagged as named entities remain intact. This replacement is done only on anchors, not on variable fragments on n-grams. As a result, extracted paraphrases cannot contain named-entity tags.

Next, we took genericizing even further by replacing anchor words by their part of speech tags. As expected, this took our approach too far; anchor text overlap became *too* common, and the results were effectively a list of the most common words in the corpora, such as "a", "the", "of", "in", "and", etc.

We also tried not replacing stop words, and only replacing non-stop words by their parts of speech,with the understanding that stop words would likely be common in both corpora. We used a relatively liberal stopword list from NLTK of 158 words. The results with this change were similar to above.

Realizing that requiring identical anchor text produces too few matches, and using part-of-speech (POS) tags or named-entity tags (NET) produces too many, we tried to find a middle road by requiring anchor similarity instead of identical anchors. For pair of anchor/variable fragments (a_1, a_2, v_1, v_2), we find a similarity metric sim(a_1, a_2), and add it to the score of (v_1, v_2). So the total score of (v_1, v_2) as a paraphrase pair is:

$$Score(v_1, v_2) = \sum_{\substack{a_1 \in v_1.\text{anchors} \\ a_1 \in v_2.\text{anchors}}} sim(a_1, a_2).$$

We define sim so that it is usually 0 in order to increase computational efficiency of our program by reducing the space of paraphrase pairs.

One possibility, based on GloVe similarity of words in the anchor, is:

$$sim(a_1, a_2) = \sum_{i=0}^{|a_1|} s_i \tag{1}$$

$$s_i = \begin{cases} 1 & \text{if } a_{1i} = a_{2i} \\ max(0.75, glv(a_1, a_2)) & \text{if } a_{1i}.\text{NET} = a_{2i}.\text{NET} \\ glv(a_1, a_2) & \text{if } glv(a_1, a_2) > 0.9 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where glv is the cosine similarity of the GloVE vectors of a_1 and a_2.

We hypothesize that many of the poor results of paraphrase extraction stem from inappropriately favoring paraphrase pairs that appear multiple times with dissimilar or trivial anchor phrases (e.g. many anchor phrases containing the word "the" in the same positions, leading to a $sim(a_1, a_2)$ of at least 1.0). We addressed this problem by increasing the GloVe cutoff to 0.9, and adding an inverse-document-frequency term to each word that we found, so that:

$$sim(a_1, a_2) = \sum_{i=0}^{|a_1|} idf(S_1, a_1) \times idf(S_2, a_2) \times s_i \tag{3}$$

$$s_i = \begin{cases} 1 & \text{if } a_{1i} = a_{2i} \\ max(0.75, glv(a_1, a_2)) & \text{if } a_{1i}.\text{NET} = a_{2i}.\text{NET} \\ glv(a_1, a_2) & \text{if } glv(a_1, a_2) > 0.9 \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Unfortunately, as summarized in Table 1, our "improvements" to anchor text overlap actually seem to greatly decrease the quality of extracted paraphrase. Furthermore, the quantity of paraphrases extracted is upper bounded by PPDB, and the fact that PPDB's paraphrases are not associated with style could be compensated for with a good algorithm for choosing the best of a set of possible paraphrases. We therefore decided to use PPDB as our source of paraphrase pairs and focus on choosing the best out of a large set of potential style translations.

4.2 Language Models

In order to capture the differences between author styles, we train character-level language models on several sources of text, each from a different author. A language model learns a probability distribution over sequences of characters. As it processes a sentence, it returns an estimate for the probability that each character will occur given all previous characters before it. Each of our language models is a LSTM recurrent neural network with two layers of 128 hidden units each using the tanh nonlinearity. The networks are trained for 50 epochs on each dataset using the Adam update rule from Kingma & Ba (2015) with a learning rate of 0.002 and cross-entropy loss. The learning rate is decayed by a factor of 2 every 5 epochs.

The trained language models can then be used to evaluate the probability of a particular sentence being produced by each author. Since the cross-entropy loss maximizes the log-likelihood of the correct character, and $P(c_1 ... c_n) =$

$\prod_{i=1}^{n} P(c_i|c_1...c_{i-1})$, we can sequentially sum the network's scores for each character in a sentence to provide a score for the sentence based on an author's language model. We can use these scores to choose between possible paraphrases generated using the methods in Section 4.1 by selecting the paraphrase with the highest score according to an author's model.

4.3 Style transfer

Once we obtain a set of paraphrase pairs, the challenge is to choose the optimal set of replacements of phrases in input sentence to maximize style transfer while retaining meaning. As a baseline, we iterate through every sentence in the text to be paraphrased. For each sentence, we generate a set of candidate paraphrases and select the paraphrase that maximizes the probability of the character sequence according to the language model of the target style. This is summarized in Algorithm 1.

Given a sentence, we compute candidate paraphrases by polling our database of extracted paraphrases (in our case from PPDB) with every word-level n-gram in the sentence for all possible values of n. For every match in the database, we add to the list of candidate paraphrases the original sentence with the match replaced with all possible paraphrase targets in the database. As an example, given the sentence "This is great", we first add the original sentence to the list of candidate paraphrases, in case it is already in the target style. We then poll PPDB with "This", "is", "great", "This is", "is great", and "This is great" and add more candidate paraphrases to the list. For example, if "great" returns "really great", "very large", and "wonderful" as some of its paraphrases, we add "This is really great", "This is very large", and "This is wonderful" to the list of candidate paraphrases. We then select the paraphrase in the list of candidate paraphrases which scores the highest according to the target style language model.

The issue with this approach is that it only performs one paraphrase replacement per sentence. With long sentences, most of the original text is left untouched, making it hard to change the style of the original sentence. To solve this issue, we split each sentence up into multiple fragments, paraphrase each fragment as before (by polling for n-gram replacements and selecting the highest-scoring replacement), and concatenate the paraphrased fragments together. Ideally, we would generate candidate paraphrases for each fragment and evaluate

Algorithm 1 Style transfer algorithm

```
Require:
  $T$, a source text
  $D$, a paraphrase database
  $F$, a sentence splitting function
  $M$, a language model for the target style
  procedure StyleTransfer(T,D,F,M):
    $T' \leftarrow$ “”                      ▷ The paraphrased text
    for each sentence $S \in T$ do
      $S' \leftarrow$ “”           ▷ The paraphrased sentence
      for each fragment $f \in F(S)$ do
        $C \leftarrow \{f\}$       ▷ The set of candidates
        for each $n$-gram $g \in f$ do
          $R \leftarrow D[g]$ ▷ Find replacements
          for $r \in R$ do
            $c \leftarrow f$ with $g$ replaced by $r$
            $C \leftarrow C \cup \{c\}$
        $f^* \leftarrow \text{argmax}_{c \in C} M(c)$
        $S' \leftarrow \text{concat}(S', f^*)$
      $T' \leftarrow \text{concat}(T', S')$
    return $T'$
```

the score of all sentence paraphrases formed by the Cartesian product of the fragments' sets of candidate paraphrases. However, this results in far too many candidate paraphrases for a given sentence. As an example, if a sentence is split into 5 fragments with 10 paraphrases each, we would have to evaluate 10^5 paraphrases for the sentence instead of 50.

We consider three approaches to splitting sentences into fragments. Each approach begins by tokenizing the sentence into a list of words; then, it decides how to break this list up into fragments. In the first and simplest approach, we split all sentences into fixed-size fragments of words. This is problematic because it ignores the structure of the sentence, possibly splitting phrases into two separate fragments. This makes the replacement of the phrase as an atomic entity impossible. It also leads to issues whenever one half of the phrase is paraphrased independently, making the final paraphrased sentence nonsensical.

A second approach is to split the sentence according to its grammatical structure as evaluated by a CKY statistical parser. Our aim was to take advantage of different phrases of a sentence, specifically verb phrases and noun phrases. We found that PPDB contained many verb phrase paraphrases, as well as a substantial number of noun phrase paraphrases. Thus, we parse each sentence into an

NLTK tree, extract noun and verb phrases into fragments such as speak to or violent conflict, and use the rest of the input sentence as fragments.

A third approach is to split the sentence at the occurrence of stop words such as and, or, that and which, as well as at punctuation like commas, semicolons, or dashes. This has the advantage of rarely splitting up phrases across fragments. However, for longer sentences without many stop words, it may not provide enough fragments. As such, it is a more conservative approach that will lead to fewer paraphrase replacements per sentence than our parse splitting approach.

To evaluate our eventual style transfer method, we implemented a simple baseline style transfer system that substitutes each word in the source text with a synonym (or itself) with the highest occurrence frequency in the target text. We used synsets from WordNet to find the set of synonyms for each word.

5 Results

5.1 Paraphrase extraction

Paraphrase extraction results are summarized in Table 1.

5.1.1 Single Corpus Paraphrase Extraction

This system was quite accurate at extracting non-trivial paraphrase pairs from both the New American Standard Bible and the Trump campaign speech dataset. The Trump dataset, while interesting, is simply not large nor repetitive enough enough to generate many interesting paraphrases other than (build a wall, have a wall) or (make America, make our country).

5.1.2 Parallel & Pseudo-Parallel Corpus Paraphrase Extraction

We adapted the algorithm to extract paraphrase pairs where the first member was from one dataset and the second member was from another dataset. We used the King James vs. New American Standard Bible as a parallel dataset. This produced accurate, but highly redundant results.

Next, we made the problem slightly harder by using a non-parallel, but topically-related dataset. The two corpora were the King James and New American versions of the bible, but using disjoint subsets of the bible from each version. Compared to the results in the parallel case, there were much fewer paraphrases generated, and the paraphrases that were noted had a lower frequency. On the plus side, the paraphrases were less obvious (e.g (fortified, fenced)), less repetitive (fewer versions of jehovah/lord), and had greater structural variation (e.g (saith jehovah of hosts, saith lord) and (land of egypt, egypt)).

5.1.3 Unrelated Non-Parallel Corpus Paraphrase Extraction

Finally, we tested the algorithm on completely non-parallel datasets, comparing the works of Dickens vs. Twain.

The algorithm was not very successful on this non-parallel datasets. Using $L_c = 3$, no paraphrase pair was observed with frequency higher than 1, and those that were observed were almost completely wrong (e.g. (little thing at, beneficial to anybody above). Using $L_c = 2$, most paraphrases were not useful (e.g. (we, i), and (i, it)), but a few synonyms ((want, wish)) were observed.

To address the bad results in the non-parallel case, we used a variety of heuristics, described in Section 4.1.1, to increase anchor-text overlap between corpora. These heuristics include transforming anchor texts into their named-entity or part-of-speech tags, and requiring similar rather than identical anchors.

These heuristics did greatly increase the number of paraphrase pairs extracted. For example, simply replacing named entities by their named-entity tags increased the number of paraphrase pairs from 68,087 to 3,351,406 for the non-parallel Bible corpus. Although the program was too slow to run the entire Dickens/Twain corpus under the anchor similarity heuristic, the number of paraphrases per sentence increased from 13 to 7148.

However, these heuristics did not significantly increase the quality of extracted paraphrases. Using named-entity recognition or part-of-speech tagging did not seem to affect the quality of extracted paraphrases; these paraphrases suffer from the same issues of incorrect or uninteresting paraphrases as the baseline results.

Using the anchor similarity heuristic, the algorithm must compare the variable fragments corresponding all pairs of anchor texts with nonzero similarity, not just identical anchor texts. As a result, the algorithm became so slow that we could only train on randomly-selected 50 sentences from each corpus. As a result, the extracted paraphrases were highly biased toward the content of those 50 sentences.

Corpus	Algorithm	# Sentences	# Anchors	# Paraphrases	Paraphrase examples: (p_1, p_2, frequency)
Trump	Single-corpus	45,257	326,420	5,606	(im, were, 10), (just want, want, 6), (our,this, 6), (our, the, 6), (make our, make this, 6), (make america, make our country, 4), (make our country, make this country, 4), (best, greatest), 4), (really want, want, 4), (china, japan, 4), (build a wall, have a wall, 4)
American Standard Bible	Single-Corpus	29,860	2,726,330	221,883	(to, unto, 146), (and the, the, 62), (that, which, 62), (god, jehovah, 60), (on, upon, 52), (thy, your, 42), (came to, shall come to, 36), (to the, unto the, 36), (jehovah, jehovah of hosts, 36), (said, saith, 32), (answered and said, said, 30), (said, said unto him, 24)
American Standard/ King James parallel	Two-corpus	29,860 34,734	2,726,330 3,301,026	624,412	(jehovah, the lord, 2337), (of jehovah, of the lord, 726), (that, which, 365), (saith jehovah, saith the lord, 226), (thy, thine, 195), (unto jehovah, unto the lord, 162), (jehovah, god, 148), (jehovah thy, the lord thy, 143), (will, shall, 143), (show, shew, 86), (my, mine, 86), (to, unto, 86)
American Standard/ King James non-parallel	Two-corpus	13,854 13,216	1,311,616 1,346,600	68,087	(jehovah, the lord, 23), (shall come to, came to, 11), (saith, said, 8), (shall come, came, 7), (of jehovah, of the lord, 6), (jehovah of hosts, the lord, 6), (twelve thousand of, and out of, 6), (jehovah of hosts the, the lord, 6), (angel, king, 5), (answered and said, said, 5)
Dickens/Twain	Two-corpus	150,802 47,072	9,392,461 3,408,523	2,727,004	(is, was, 257), (i, he, 157), (i, we, 78), (she, he, 74), (you, i, 68), (am, was, 64), (i, they, 52), (there, it, 49), (he, we, 49), (have, had, 45), (they, he, 43), (can, could, 42), (i, she, 42), (should, would, 39), (was not, was, 39), (has, had, 36), (had, was, 36), (dont, do not, 34)
Dickens/Twain	Two corpus + stemming	140,858 46170	6276552 2271702	1443058	(i, he, 114), (he, i, 73), (is, wa, 66), (i, we, 62), (you, i, 58), (she, he, 53), (am, wa, 38), (can, could, 31), (she, they, 29), (they, he, 29), (wa, am, 28), (you, we, 28), (wish, want, 27), (we, he, 24), (did, do, 24), (he, you, 24), (you, they, 23), (know, think, 23), (dont, do not, 23)
Dickens/Twain	Two corpus + NER	150,802 47,072	9,318,069 3,397,346	437,511	and, said, 622), (and, and mr., 316), (not, n't), 308), (was, is, 293), (is, was, 229), (he, i, 197), ((n't, not, 186), (and, or, 160), (did, do, 154), (and, of, 151), (i, he, 136), (john, and, 133), (you, i, 116), (and, esquire, 112), (we, i, 109), (he, she, 94),(and, and mr, 81), (sir, mr., 79)
American Standard/ King James non-parallel	Two corpus + NER	29,782, 22,197	1,580,464 1,225,125	3,351,406	(begat, and, 18143), (son, daughter, 13238), (daughter, son, 12303), (and, begat, 12119), (sons, son, 6116), (children, son, 5942), (god, son, 5857), (son, king, 5213), (son, sons, 5051), (son, father, 4783), (king, son, 4272), (son, wife, 3605), (brother, son, 3360), (jehovah, the lord, 3096)
Dickens/Twain	Two-corpus + POS tags	5000 5000	49,376 49,117	9,471,842	(a, the, 41168), (the, a, 37719), (and, of, 21597), (the, his, 19742), (of, in, 18728), (in, of, 18571), (of, and, 17658), (his, the, 10161), (of, with, 9523), (a, his, 9267), (the, her, 7671), (of, on, 7197), (and, in, 6883), (of, to, 6868),(of, at, 6326), (the, my, 6278), (for, of, 6155), (with, of, 5600)
Dickens/Twain	POS tags, stopwords intact	5000 5000	166883 146570	941,717	(and, of, 5113), (of, and, 4481), (the, a, 4392), (of, in, 4058), (a, the, 4057), (in, of, 3560), (of, on, 1748), (of the, of, 1488), (of, with, 1474), (and, in, 1411), (his, the, 1282), (of, at, 1274) (of, to, 1158), (of, of the, 1150), (to, of, 1069), (in, and, 1055), (a, his, 919), (on, of, 917), (the, her, 909)
Dickens/Twain	GloVe Similarity, Threshold=0.75	50 50	2738 2287	610,155	(a, the, 190.6), (and, the, 175.0), (the, i, 153.6), (and, in, 142.2), (the, to, 130.8), (and, i, 129.5), (a, i, 128.5), (and, to, 12.72), (the, and, 123.3), (the, a, 109.0), (the, in, 104.9), (to, the, 97.9) (i, the, 86.60), (and, that, 84.2), (a, to, 81.3), (and, with, 79.8), (a, and, 78.3), (a, in, 78.1)
Dickens/Twain	GloVe Similarity+IDF, Threshold=0.9	50 50	3217 2414	714,833	(had, i, 4.6), (i, place, 4.5), (left, at, 4.5), (i left, place at, 4.5), (i, took place, 4.5), (had disappeared at, i, 4.5), (i, interview took place, 4.5), (had disappeared, i, 4.5), (when i, interview took place, 4.5), (it had, where i had, 4.4), (had disappeared at, i had passed, 4.4)

Table 1: Anchor-based paraphrase extraction results. In the second column, "NER" indicates that anchor words are transformed into their named-entity tag, if one exists. "POS tags" indicates that anchor words are transformed into their part-of-speech tag. "POS tags, stop words intact" indicate that all anchor words except for stop words have been transformed into their POS tag. "GloVe Similarity" means that GloVe similarity was used for anchor matching as in Eq. 3. "Threshold" refers for the lowest GloVe similarity that can contribute to a similarity score. "IDF" indicates that IDF discounting is used as in Eq. 4.

5.2 Language models

To verify that our language models learn the stylistic subtleties of each author, we generate new text by sampling one character at a time from the network's learned probability distribution. The following are samples generated by our language models:
Donald Trump:

> We have to do this. We are going to be a great deal. But I want to thank the best, and they said I want to take a lot of the military, and it's a good better than anything to win a lot of millions of money and some of the best and they have to pay to that was a great campaign

William Shakespeare:

> I will fair and be born. The play is a beast of my wise in the honor in a fair of his some hear the words. What thou mad to your grace in the command, for his news more will go do the return to do your courts. . . .

Overall, the LSTM language models seem to capture the style of their intended authors. Since we use these language models to choose between possible paraphrases, we also compared the scores of various phrases under each style's language model to verify our assumptions about which phrases are more likely in each style. Table 2 shows example paraphrase scores for each language model. The language model scores highest for the paraphrase which most closely resembles the style of the language model's author.

5.3 Style transfer

We compare the results of our style transfer approaches in Table 3. For our evaluation, we chose

Phrase	Trump	Twain	Shakesp
You are really tremendous	**6.485**	5.555	4.723
You are beautiful	5.175	**5.728**	4.911
Thou art lovely	4.515	4.416	**5.238**

Table 2: Raw scores for various phrases under language models trained on Trump, Twain, and Shakespeare corpora.

Transformation	Trump	Twain	Shakesp
Original	6.252	6.523	5.684
Baseline synonym replacement	6.242	6.469	5.707
Parse splitting + paraphrasing	6.556	6.656	5.843
Stopword splitting + paraphrasing	**6.653**	**6.753**	**5.864**

Table 3: Raw scores for Martin Luther King Jr.'s "I Have a Dream" speech before and after different style transfer approaches under each style's language model, using PPDB as the paraphrase source.

Martin Luther King Jr's "I Have a Dream" speech because it is written in modern English and contains a variety of literary devices.

Our naive baseline style transfer approach using synonym substitution performs poorly, as expected. Since the baseline approach does not rely on the language model of the target author (rather, it relies on their word usage frequency), the scores are barely improved, if at all. Frequently, words had no synonyms in WordNet, or synonyms were substituted that didn't preserve the original meaning of the word, such as "queen" to "king" and "father" to "mother". We found that grammar and syntactical correctness suffer, and style transfer is rarely observed since replacements based on frequency favor replacements like "was" to "be", which are a feature of the English language and not characteristic of any speaker's style.

We found that splitting sentences along noun and verb phrases using a sentence parse had comparable results to splitting at the occurrence of stop words, although splitting at stop words seemed to result in consistently higher language model scores. Because of its more fine-grained nature, splitting according to the parse tree resulted in more fragments per sentence and thus more paraphrase substitutions (see Table 4), losing more of the original meaning of the sentence. Ideally, more substitutions would capture more of the target style, presenting a trade-off between preservation of original meaning and style transfer. Unfortunately, grammar and syntactical meaning also suffer from too many paraphrase substitutions, suggesting that parse splitting is sub-optimal and that the more conservative stop word splitting approach may work better in practice.

	None	Stopword	Parse
Fragments per sentence	1.00	2.84	6.21
Candidates per fragment	83.25	29.84	16.44
Candidates per sentence	83.25	84.61	102.10

Table 4: Average statistics of different sentence fragmentation methods on "I Have a Dream".

Examples of style-transferred sentences using various methods and target styles are shown in Table 5. Interesting replacements include: changing "justice" to "administration of justice" (Trump), "the judge" (Twain), and "the courts" (Shakespeare); "check" to "cheque" (Shakespeare); "It is obvious" to "That's obvious" (Trump) and "It is clearly evident" (Twain); and "promise" to "not promising anything" (Trump). However, many of the generated candidate paraphrases are grammatically incorrect or change the meaning of the original sentence. While the first issue can be mitigated by filtering sentences that cannot be parsed, the second is more problematic. Ideally, incorrect paraphrases could be fixed with a more accurate paraphrase databases, but in many cases the validity of a paraphrase replacement is dependent on sentence structure and context. Another visible issue with our approach is that even if our chosen replacement is valid, it does not always result in recognizably-transferred style. We believe this is because an important source of style is derived from sentence structure, which our approach does not modify significantly. While the language model is capable of selecting the paraphrase which more closely resembles the typical structure in the target style, our phrase replacement approach can only generate candidates which modify sentence struc-

Transformation	Trump	Twain	Shakespeare
Baseline synonym replacement	It **be** obvious today that America **have** defaulted on this promissory note.	It **be** obvious today that America **have** **default** on this promissory note.	It **be** obvious today that America **have** defaulted on this promissory note.
	And so, we've **do** to cash this check, a **see** that will give us upon **need** the **wealth** of freedom and the security of justice.	And so, we've **do** to cash this check, a **see** that will give us upon **need** the **wealth** of freedom and the security of justice.	And so, we've **do** to cash this check, a **see** that will give us upon **need** the **wealth** of freedom and the security of justice.
	Now **be** the time to **have very** the **promise** of democracy.	Now **be** the time to **have very** the **promise** of democracy.	Now **be** the time to **have very** the **promise** of democracy.
Parse splitting + paraphrasing	**That's** obvious today that America has defaulted **with respect to** this promissory note.	It is **clearly evident the present day** that America has defaulted **with respect to** this promissory note.	It is **very** obvious **the present day** that America has defaulted **with respect to** this promissory note.
	And so, **we do have** come to cash this check, a check that will give **the United States** upon demand the riches of **their liberty** and the security of justice.	And **well, then,** we've come to cash this check, a check that will give **the United States** upon **request** the riches of freedom and the security of justice.	And so, we **do have** come to cash this check, a check that will give us upon demand the riches of **their liberty** and the security of justice.
	Now **that's the time to be doing** real the **'m not promising anything** of **democratic politics**.	Now is the good time to **be done very** real the promises of **democratic politics**.	Now is the **high** time to **are doing very** real the promises of democracy.
Stopword splitting + paraphrasing	**That's** obvious today that America has defaulted **with respect to** this promissory note.	It is **clearly evident** today that America has defaulted **with respect to** this promissory note.	It is obvious **the present day** that America has defaulted **with respect to** this promissory note.
	And **well, then,** we've come to **the money** this check, **control** that will give us upon **request** the riches of freedom and the security **of administration** of justice.	And **well, then,** we've come to **the money** this check, **control** that will give us upon **request** the riches of freedom and the security of **the judge**.	And **well, then,** we **do have** come to cash this check, a **cheque** that will give us upon **request** the riches of freedom and the security of **the courts**.
	Now **that's** the time to make real the promises of democracy.	Now **that's** the time to make real the promises of democracy.	Now is the time to make **true** the promises of democracy.

Table 5: Style-transfer paraphrases for "It is obvious today that America has defaulted on this promissory note" (top), "And so, we've come to cash this check, a check that will give us upon demand the riches of freedom and the security of justice" (middle), and "Now is the time to make real the promises of democracy" (bottom) under each language model using PPDB as the paraphrase source.

ture by expanding or reducing clauses, never adding or removing clauses or rearranging their order.

6 Conclusion and Future Work

We explored style transfer based on an algorithm with two steps: paraphrase extraction, and paraphrase replacement using language models. In our paraphrase extraction work, we were unable to overcome the challenge of a small amount of anchor-text overlap between the two corpora, despite using part-of-speech and named-entity tags to our advantage. Instead, we used the PPDB paraphrase database as a source of paraphrases. However, even with access to a large paraphrase database, using extracted paraphrases to transform text is nontrivial, and the task of choosing which paraphrases to use is delicate and not always successful.

As a simple improvement to our work, paraphrase candidates for sentences which do not have a plausible syntactic parse could be filtered out. Also, rather than selecting the highest scoring fragment paraphrases independently, we could select the highest scoring first fragment, and then seed the language model with the paraphrased fragment before evaluating the next fragment's candidates, and so on. This would hopefully fix issues where performing multiple replacements renders the structure of the sentence invalid or incoherent.

Future approaches could explore less rigid paraphrase approaches that rely on restructuring the syntax of the sentence rather than performing per-fragment replacements. Another idea is bootstrapping: after generating some set of paraphrases, we can replace the source corpus with a paraphrased version, thereby making the source corpus more similar to the destination corpus. This would hopefully generate a larger set of paraphrases. Evaluating style transfer models quantitatively by asking humans to rank style-transferred sentences would also be worth exploring.

References

[Barzilay & Lee 2003] R. Barzilay, L. Lee 2003. *A Neural Probabilistic Language Model*. Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology, 16–23. aclweb.org/anthology/N/N03/N03-1003.pdf.

[Barzilay & McKeown 2001] R. Barzilay, K.R. McKeown 2001. *Extracting Paraphrases from a Parallel Corpus*. Proceedings of the 39th Annual Meeting on Association for Computational Linguistics, 50–57. www.aclweb.org/anthology/P01-1008.

[Christian Classics Ethereal Library 2015] H. Plantinga 2015. http://www.ccel.org/ccel/bible.

[Gatys et al. 2015] L. A. Gatys, A. S. Ecker, M. Bethge1 2015. *A Neural Algorithm of Artistic Style*. CoRR 2015, http://arxiv.org/pdf/1508.06576v2.pdf

[Kingma & Ba 2015] D. Kingma, J. Lei Ba 2015. *Adam: A Method for Stochastic Optimization*. ICLR 2015, http://arxiv.org/pdf/1412.6980v8.pdf

[Pasca & Dienes 2005] M. Pasca, P. Dienes 2005. *Aligning Needles in a Haystack: Paraphrase Acquisition Across the Web*. Natural Language Processing IJCNLP 2005, 119–130. aclweb.org/anthology/I05-1011

[PPDB 2013] J. Ganitkevitch, B. Van Durme, C. Callison-Burch 2013. *PPDB: The Paraphrase Database*. Proceedings of NAACL-HLT, 758–764. http://www.cis.upenn.edu/ ccb/ppdb/

[Project Gutenberg 2016] *Free eBooks by Project Gutenberg*. 2016 https://www.gutenberg.org/.

[WhatTheFolly.com 2016] . *What The Folly?! Donald Trump speech transcripts*. 2016. http://www.whatthefolly.com/tag/donald-trump/.

[Xu 2012] W. Xu, A. Ritter, B. Dolan, R. Grishman, C. Cherry 2012. *Paraphrasing for Style*. COLING, 2899–2914. http://aclweb.org/anthology/C/C12/C12-1177.pdf.