Shakespearizing Modern Language Using Copy-Enriched Sequence-to-Sequence Models

Harsh Jhamtani ; Varun Gangal ; Eduard Hovy, Eric Nyberg
Language Technology Institute
Carnegie Mellon University
{jharsh,vgangal,hovy,ehn}@cs.cmu.edu

Abstract

Variations in writing styles are commonly used to adapt the content to a specific context, audience, or purpose. However, applying stylistic variations is still by and large a manual process, and there have been little efforts towards automating it. In this paper we explore automated methods to transform text from modern English to Shakespearean English using an end to end trainable neural model with pointers to enable copy action. To tackle limited amount of parallel data, we pre-train embeddings of words by leveraging external dictionaries mapping Shakespearean words to modern English words as well as additional text. Our methods are able to get a BLEU score of 31+, an improvement of \( \approx 6 \) points above the strongest baseline.

1 Introduction

Text is often morphed using a variety of lexical and grammatical transformations, adjusting the degree of formality, usage of catchy phrases, and other such stylistic changes to make it more appealing. Moreover, different text styles appeal to different user segments (Saha Roy et al., 2015) (Kitis, 1997) (Schwartz et al., 2013). Thus there is a need to quickly adapt text to different styles. However, manually transforming text to desired style can be a tedious process.

There have been increased efforts towards machine assisted text content creation and editing through automated methods for summarization (Rush et al., 2015), brand naming (Hiranandani et al., 2017), text expansion (Srinivasan et al., 2017), etc. However, there is a dearth of automated solutions for adapting text quickly to differ-

| No | Type | Text |
|----|------|------|
| 1  | MODERN | Oh my, my bones ache so much |
|    | ORIGINAL | For, how my bones ache! |
|    | COPY | you'll be, sir, what the bones are tured. |
|    | STAT | Oh my, my bones ache so much. |
| 2  | MODERN | I am in a rush. |
|    | ORIGINAL | I stand on sudden haste. |
|    | COPY | I'm in a hurry. |
|    | STAT | I am in sudden haste. |
| 3  | MODERN | I commend to thy lady. |
|    | ORIGINAL | I commend to your lady. |
|    | COPY | I commend to your lady. |
|    | STAT | I commend to your lady. |
| 4  | MODERN | Well, sir, my mistress is the sweetest lady. |
|    | ORIGINAL | Well sir, my mistress is the sweetest lady. |
|    | COPY | Well sir, my mistress is the sweetest lady. |
|    | STAT | Well sir, my mistress is the sweetest lady. |
| 5  | MODERN | Showing mercy by pardoning killers only causes more murders. |
|    | ORIGINAL | Mercy but murders, pardoning those that kill. |
|    | COPY | mercy but murders, those who kill us. |
|    | STAT | mercy but murders, those who kill us. |
| 6  | MODERN | Holy Saint Francis, this is a drastic change! |
|    | ORIGINAL | Holy Saint Francis, what a change is here? |
|    | COPY | it's the holy flute, what's the changed! |
|    | STAT | Holy Saint Francis, this is a drastic change! |
| 7  | MODERN | Was that my father that went hence so fast? |
|    | ORIGINAL | Was that my father that left here in such a hasty? |
|    | COPY | was that my father that went hence so fast? |
|    | STAT | was that my father that left here in such a hasty? |
| 8  | MODERN | Give me one kiss and I'll go down. |
|    | ORIGINAL | One kiss, and I'll descend. |
|    | COPY | one kiss, and I'll descend. |
|    | STAT | Give me one kiss, and I'll go down. |
| 9  | MODERN | Then the window will let day in, and life goes out the window. |
|    | ORIGINAL | Then, window, let day in and life out. |
|    | COPY | then, window, out and day life. |
|    | STAT | then the window will let day in, and life out. |

Table 1: Examples from dataset showing modern paraphrases (MODERN) of few sentences from Shakespeare’s plays (ORIGINAL). We also show transformation of modern text to Shakespearean text from our models (COPY, SIMPLES2S and STAT).
ent styles. We consider the problem of transforming text written in modern English text to Shakespearean style English. For the sake of brevity and clarity of exposition, we henceforth refer to the Shakespearean sentences/side as Original and the modern English paraphrases as Modern.

Unlike traditional domain or style transfer, our task is made more challenging by the fact that the two styles employ diachronically disparate registers of English - one style uses the contemporary language while the other uses Early Modern English \(^1\) from the Elizabethan Era (1558-1603). Although Early Modern English is not classified as a different language (unlike Old English and Middle English), it does have novel words (acknown and belike), novel grammatical constructions (two second person forms - thou (informal) and you (formal) (Brown et al., 1960)), semantically drifted senses (e.g fetches is a synonym of excuses) and non-standard orthography (Rayson et al., 2007). Additionally there is a domain difference since the Shakespearean play sentences are from a drama whereas the parallel modern English sentences are meant to be simplified explanation for high-school students.

Prior works in this field leverage language model for target style, achieving transformation either using phrase tables (Xu et al., 2012), or by inserting relevant adjectives and adverbs (Saha Roy et al., 2015). Such works have limited accuracy and scope in the type of transformations that can be achieved. Moreover, statistical and rule MT based systems do not provide a direct mechanism to a) share word representation information between source and target sides b) incorporating constraints between words into word representations in end-to-end fashion. Neural sequence-to-sequence models, on the other hand, provide such flexibility (Faruqui et al., 2014).

Our main contributions are as follows:

- We use a sentence level sequence to sequence neural model with a pointer network component to enable direct copying of words from input. We demonstrate that this method performs much better than prior phrase translation based approaches for transforming Modern English text to Shakespearean English.

- We pre-train word embedding considering external dictionary of words. The pretrained embedding enable our model to learn to transform text from small amount of parallel data.

Rest of the paper is organized as follows. We first provide a brief analysis of our dataset in (§2). We then elaborate on details of our methods in (§3, §4, §5, §6). We then discuss experimental setup and baselines in (§7). Thereafter, we discuss the results and observations in (§8). We conclude with discussions on related work (§9) and future directions (§10).

### 2 Dataset

Our dataset is a collection of line-by-line modern paraphrases for 16 of Shakespeare’s 36 plays (Antony & Cleopatra, As You Like It, Comedy of Errors, Hamlet, Henry V etc) from the educational site Sparknotes\(^2\). This dataset was compiled by Xu et al. (2014; 2012) and is freely available on github.\(^3\) 14 plays covering 18,395 sentences form the training data split. We kept 1218 sentences from the play Twelfth Night as validation data set. The last play, Romeo and Juliet, comprising of 1462 sentences, forms the test set.

#### 2.1 Examples

Table 1 shows some parallel pairs from the test split of our data, along with the corresponding target outputs from some of our models. Copy and SimpleS2S refer to our best performing attentional S2S models with and without a Copy component respectively. Stat refers to the best statistical machine translation baseline using off-the-shelf GIZA++ aligner and MOSES. We can see through many of the examples how direct copying from the source side helps the Copy generates better outputs than the SimpleS2S. The approaches are described in greater detail in (§3) and (§7).

#### 2.2 Analysis

Table 2 shows some statistics from the training split of the dataset. In general, the Original side

|                  | Original | Modern |
|------------------|----------|--------|
| # Word Tokens    | 277K     | 200K   |
| # Word Types     | 12.39K   | 10.05K |
| Average Sentence Length | 11.81    | 10.91  |
| Trigram Type Dist | 6.06     | 0.8   |
| 1-gram Word Types | 6.15     | 6.06  |

Table 2: Dataset Statistics

---

1\(^\text{http://tinyurl.com/otuqynl}\)

2\(^\text{www.sparknotes.com}\)

3\(^\text{http://tinyurl.com/ycdd3v6h}\)
has longer sentences and a larger vocabulary. The slightly higher entropy of the Original side’s frequency distribution indicates that the frequencies are more spread out over words. Intuitively, the large number of shared word types indicates that this sharing the representation between Original and Modern sides could provide some benefit.

3 Method Overview

Overall architecture of the system is shown in Figure (§1). We use a bidirectional LSTM to encode the input modern English sentence. Our decoder side model is a mixture model of RNN module and pointer network module. The two individual modules share the attentions weights over encoder states, although it is not necessary to do so. The decoder RNN predicts probability distribution of next word over the vocabulary, while pointer model predicts probability distribution over words in input. The two probabilities undergo a weighted addition, the weights themselves computed based on previous decoder hidden state and the encoder outputs.

Let $x, y$ be the some input - output sentence pair in the dataset. Both input $x$ as well as output $y$ are sequence of tokens. $x = x_1 x_2 \ldots x_{|T_{enc}|}$, where $T_{enc}$ represents the length of the input sequence $x$. Similarly, $y = y_1 y_2 \ldots y_{T_{dec}}$. Each of $x_i, y_j$ is be a token from the vocabulary.

4 Token embeddings

Each token in vocabulary is represented by a $M$ dimensional embedding vector. Let vocabulary $V$ be the union of modern English and Shakespearean vocabularies i.e. $V = V_{modern} \cup V_{shakespeare}$. $E_{enc}$ and $E_{dec}$ represent the embedding matrices used by encoder and decoder respectively ($E_{enc}, E_{dec} \in \mathbb{R}^{V \times M}$). We consider union of the vocabularies for both input and output embeddings because many of the tokens are common in two vocabularies, and in the best performing setting we share embeddings between encoder and decoder models. Let $E_{enc}(t)$, represent encoder side embeddings of some token $t$. For some input sequence $x$, $E_{enc}(x)$ is given as $(E_{enc}(x_1), E_{enc}(x_2), \ldots)$.

4.1 Pretraining of embeddings

Learning token embeddings from scratch in an end-to-end fashion along with the model greatly increases the number of parameters. To mitigate this, we consider pretraining of the token embeddings. We pretrain our embeddings on all training sentences. We also experiment with adding additional data from PTB (Marcus et al., 1993) for better learning of embeddings. Additionally we leverage a dictionary mapping tokens from Shakespearean English to modern English.

We consider four distinct strategies to train the embeddings. In the cases where we use external text data, we first train the embeddings using both the external data and training data, and then for the same number of iterations on training data alone, to ensure adaptation. Note that we do not directly use off-the-shelf pretrained embeddings such as Glove (Pennington et al., 2014) and Word2Vec (Mikolov et al., 2013) since we need to learn embeddings for novel word forms (and also different word senses for extant word forms) on the Original side.

4.1.1 Plain

This method is the simplest pre-training method. Here, we do not use any additional data, and train word embeddings are trained on the union of Modern and Original sentences.

4.1.2 PlainExt

In this method, we add all the sentences from the external text source (PTB) in addition to sentences in training split of our data.

4.1.3 Retro

We leverage a dictionary $L$ of approximate Original $\rightarrow$ Modern word pairs (Xu et al., 2012; Xu, 2014), crawled from shakespeare-words.com, a source distinct from Sparknotes. We explicitly add the two 2nd persons and their corresponding forms (thy, thou, thyself etc) which are very frequent but not present in $L$. The final dictionary we use has 1524 pairs. Faruqui et al (2014) proposed a retrofitting method to update a set of word embeddings to incorporate pairwise similarity constraints. Given a set of embeddings $p_i \in P$, a vocabulary $V$, and a set $C$ of pairwise constraints $(i, j)$ between words, retrofitting tries to learn a new set of embeddings $q_i \in Q$ to minimize the following objective:

$$f(Q) = \delta \sum_{i=1}^{|V|} (p_i - q_i)^2 + \omega \sum_{(i,j) \in C} (q_i - q_j)^2$$  \hspace{1cm} (1)
We use their off-the-shelf implementation \(^4\) to encode the dictionary constraints into our pretrained embeddings, setting \(C = L\) and using suggested default hyperparameters for \(\delta\), \(\omega\) and number of iterations.

4.1.4 RetroExt

This method is similar to Retro, except that we use sentences from the external data (PTB) in addition to training sentences.

We use None to represent the settings where we do not pretrain the embeddings.

4.2 Fixed embeddings

Fine-tuning pre-trained embeddings for a given task may lead to overfitting, especially in scenarios with small amount of supervised data for the task (Madhyastha et al., 2015). This is because embeddings for only a fraction of vocabulary items get updated, leaving the embeddings unchanged for many vocabulary items. To avoid this, we consider fixed embeddings pretrained as per procedures described earlier. While reporting results in Section (§8), we separately report results for fixed (FIXED) and trainable (VAR) embeddings, and observe that keeping embeddings fixed leads to better performance.

5 Method Description

In this section we give details of the various modules in the proposed neural model.

\(^4\)github.com/mfaruqui/retrofitting

5.1 Encoder model

Let \(\text{LSTM}_{\text{enc}}\) and \(\text{LSTM}_{\text{enc}}\) represent the forward and reverse encoder. \(h_t^{\text{enc}}\) represent hidden state of encoder model at step \(t\) \((h_t^{\text{enc}} \in \mathbb{R}^H)\). The following equations describe the model:

\[
\begin{align*}
    h_0^{\text{enc}} &= \overrightarrow{0}, \ h_1^{\text{enc}} = \overrightarrow{0} \quad (2) \\
    h_t^{\text{enc}} &= \text{LSTM}_{\text{enc}}\left(h_{t-1}^{\text{enc}}, E_{\text{enc}}(x_t)\right) \quad (3) \\
    h_t^{\text{enc}} &= \text{LSTM}_{\text{enc}}\left(h_{t}^{\text{enc}}, E_{\text{enc}}(x_t)\right) \quad (4) \\
    h_t^{\text{enc}} &= h_t^{\text{enc}} + h_t^{\text{enc}} \quad (5)
\end{align*}
\]

We use addition to combine the forward and backward encoder states, rather than concatenation which is standardly used, since it doesn’t add extra parameters, which is important in a low-data scenario such as ours.

5.2 Attention

Let \(h_t^{\text{dec}}\) represent the hidden state of the decoder LSTM at step \(t\). Let \(E_{\text{dec}}(y_{t-1})\) represent the decoder side embeddings of previous step output. We use special \(\text{START}\) symbol at \(t = 1\).

We first compute a query vector, which is a linear transformation of \(h_{t-1}^{\text{dec}}\). A sentinel vector \(s \in \mathbb{R}^H\) is concatenated with the encoder states to create \(F_{\text{att}} \in \mathbb{R}^{(T_{\text{enc}}+1) \times H}\), where \(T_{\text{enc}}\) represents the number of tokens in encoder input sequence \(x\). A normalized attention weight vector \(\alpha_{\text{norm}}\) is computed. The value \(g\), which corresponds to attention weight over sentinel vector, represents the
weight given to the decoder RNN module while computing output probabilities.

\[ q = h^{dec}_{t-1} W_q \]

\[ F_{att} = \text{concat}(h^{enc}_{t:T_{enc}}, s) \]

\[ \alpha_i = \sum_{j=1}^{H} \text{tanh}(F_{att}^{(j)} q_i) + b, \quad \alpha_i, b \in \mathbb{R} \]

\[ \alpha^{norm} = \text{softmax}(\alpha) \]

\[ \beta = \alpha^{norm} \in \mathbb{R}^{T_{enc}+1} \]

\[ g = \alpha^{norm}_{T_{enc}+1} \in \mathbb{R} \]

5.3 Pointer model

As pointed out earlier, a pair of corresponding Original and Modern sentences have significant vocabulary overlap. Moreover, there are lot of proper nouns and rare words which might not be predicted by a sequence to sequence model. To rectify this, pointer networks have been used to enable copying of tokens from input directly (Merity et al., 2016). The pointer module provides location based attention, and output probability distribution due to pointer network module can be expressed as follows:

\[ P^{PTR}_t(w) = \sum_{x_j = w} (\beta_j) \]  

5.4 Decoder RNN

Summation of encoder states weighed by corresponding attention weights yields context vector. Output probabilities over vocabulary as per the decoder LSTM module are computed as follows:

\[ c_t = \sum_{i=1}^{T_{enc}} \beta_i h^{enc}_i \]

\[ h^{dec}_t = LSTM(h^{dec}_{t-1}, [\text{concat}(E_{dec}(y_{t-1}), c_t)]) \]

\[ P^{LSTM}_t = \text{softmax}(W_{out} [\text{concat}(h^{dec}_t, c_t)] + b^{out}) \]

During training, we feed the ground truth for \( y_{t-1} \), whereas while making predictions on test data, predicted output from previous step is used instead.

5.5 Output prediction

Output probability of a token \( w \) at step \( t \) is a weighted sum of probabilities from decoder LSTM model and pointer model given as follows:

\[ P_t(w) = g \times P^{LSTM}_t(w) + (1-g) \times P^{PTR}_t(w) \]  

\[ P^{PTR}_t(w) \text{ takes a non-zero value only if } w \text{ occurs in input sequence, otherwise it is } 0. \]

Cross entropy loss is used to train the model. For a data point \( (x, y) \in D \) and predicted probability distributions \( P_t(w) \) over the different words \( w \in V \) for each time step \( t \in \{1, \ldots, T_{dec}\} \), the loss is given by

\[ -\sum_{t=1}^{T_{dec}} \log(P_t(y_t)) \]  

Sentinel Loss (SL): Following from work by (Merity et al., 2016), we consider additional sentinel loss. This loss function can be considered as a form of supervised attention. Sentinel loss is given as follows:

\[ -\sum_{t=1}^{T_{dec}} \log(q^{(i)} + \sum_{x_j = y_i} (\beta_j^{(i)})) \]  

We report the results demonstrating the impact of including the sentinel loss function (+SL).

7 Experiments

In this section we describe the experimental setup and evaluation criteria used.

7.1 Preprocessing

We lowercase sentences and then use NLTK’s PUNKT tokenizer to tokenize all sentences. The Original side has certain characters like æ which are not extant in today’s language. We map these characters to the closest equivalent character(s) used today (e.g æ → ae)

7.2 Baseline Methods

7.2.1 As-it-is

Since both source and target side are English, just replicating the input on the target side is a valid and competitive baseline, with a BLEU of 21+. 
7.2.2 Dictionary

Xu et al. (2012) provide a dictionary mapping between large number of Shakespearean and modern English words. We augment this dictionary with pairs corresponding to the 2nd person thou (thou, thy, thyself) since these common tokens were not present.

Directly using this dictionary to perform word-by-word replacement is another admissible baseline. As was noted by Xu et al. (2012), this baseline actually performs worse than As-it-is. This could be due to its performing aggressive replacement without regard for word context. Moreover, a dictionary cannot easily capture one-to-many mappings as well as long-range dependencies.

7.2.3 Off-the-shelf SMT

To train statistical machine translation (SMT) baselines, we use publicly available open-source toolkit MOSES (Koehn et al., 2007), along with the GIZA++ word aligner (Och, 2003), as was done in (Xu et al., 2012). For training the target-side LM component, we use the lmplz toolkit within MOSES to train a 4-gram LM. We also use MERT (Och, 2003), available as part of MOSES, to tune on the validation set.

For fairness of comparison, it is necessary to use the pairwise dictionary and PTB while training the SMT models as well - the most obvious way for this is to use the dictionary and PTB as additional training data for the alignment component and the target-side LM respectively. We experiment with several SMT models, ablating for the use of both PTB and dictionary. In 8, we only report the performance of the best of these approaches, with more detailed results covered in Appendix.

7.3 Evaluation

Our primary evaluation metric is BLEU (Papineni et al., 2002). We compute BLEU using the freely available and very widely used perl script from the MOSES decoder.

We also report PINC (Chen and Dolan, 2011), which originates from paraphrase evaluation literature and evaluates how much the target side paraphrases resemble the source side. Given a source sentence s and a target side paraphrase c generated by the system, PINC(s, c) is defined as:

\[
PINC(s, c) = 1 - \frac{1}{N} \sum_{n=1}^{N} \frac{|Ngram(c, n) \cap Ngram(s, n)|}{|Ngram(c, n)|}
\]

where \(Ngram(x, n)\) denotes the set of n-grams of length \(n\) in sentence \(x\), and \(N\) is the maximum length of n-gram considered. We set \(N = 4\). Higher the PINC, greater the novelty of paraphrases generated by the system. Note, however, that PINC does not measure fluency of generated paraphrases.

7.4 Training and Parameters

We use a minibatch-size of 32 and the ADAM optimizer (Kingma and Ba, 2014) with learning rate 0.001, momentum parameters 0.9 and 0.999, and \(\epsilon = 10^{-8}\). All our implementations are written in Python using Tensorflow 1.1.0 framework.

For every model, we experimented with two configurations of embedding and LSTM size - \(S\) (128-128), \(ME\) (192-192) and \(L\) (256-256). Across models, we find that the \(ME\) configuration performs better in terms of highest validation BLEU. We also find that larger configurations (384-384 & 512-512) fail to converge or perform very poorly. Here, we report results only for the \(ME\) configuration for all the models. For all our models, we picked the best saved model over 15 epochs which has the highest validation BLEU.

7.5 Decoding

At test-time we use greedy decoding to find the most likely target sentence. We also experiment with a post-processing strategy which replaces UNKs in the target output with the highest aligned (maximum attention) source word. We find that this gives a small jump in BLEU of about 0.1-0.2 for all neural models. Our best model, for instance, gets a jump of 0.14 to reach a BLEU of 31.26 from 31.12.

8 Results

The results in Table 3 confirm most of our hypotheses about the right architecture for this task.

• Copy component: We can observe from Table 3 that the various Copy models each

\[\text{http://tinyurl.com/yben45gm}\]

\[\text{\textcopyright\textsuperscript{\textregistered} 2023 \textregistered}\]
outperform their SimpleS2S counterparts by at least 7-8 BLEU points.

- **Retrofitting dictionary constraints:** The Retro configurations generally outperform their corresponding Plain configurations. For instance, our best configuration Copy.Yes.RetroExtFixed gets a better BLEU than Copy.Yes.PlainExtFixed by a margin of at least 11.

- **Sharing Embeddings:** Sharing source and target side embeddings benefits all the Retro configurations, although it slightly deteriorates performance (about 1 BLEU point) for some of the Plain configurations.

- **Fixing Embeddings:** Fixed configurations always perform better than corresponding Var ones (save some exceptions). For instance, Copy.Yes.RetroExtFixed gets a BLEU of 31.12 compared to 20.95 for Copy.Yes.RetroExtVar. Due to fixing embeddings, the former has just half as many parameters as the latter (5.25M vs 9.40M).

- **Effect of External Data:** Pretraining with external data Ext works well along with retrofitting Retro. For instance, Copy.Yes.RetroExtFixed gets a BLEU improvement of 2+ points over Copy.Yes.RetroFixed

- **Effect of Pretraining:** For the SimpleS2S models, pre-training adversely affects BLEU. However, for the Copy models, pre-training leads to improvement in BLEU. The simplest pretrained Copy model, Copy.No.PlainVar has a BLEU score 1.8 higher than Copy.No.NoneVar.

- **PINC scores:** All the neural models have higher PINC scores than the statistical and dictionary approaches, which indicate that the target sentences produced differ more from the source sentences than those produced by these approaches.

- **Sentinel Loss:** Adding the sentinel loss does not have any significant effect, and ends up reducing BLEU by a point or two, as seen with the Copy+SL configurations.

### Table 3: Test BLEU results.

| Model     | Sh | Init | BLEU (PINC) |
|-----------|----|------|-------------|
| AS-IT-IE  | -  | -    | 21.73 (79.09) |
| DICTIONARY | ✓  | -    | 21.20 (81.61) |
| AS-IT-IE  | -  | ✓    | 20.26 (81.54) |
| SIMPLES2S | ✓  | ✓    | 21.13 (79.92) |
| SIMPLES2S | ✓  | ✓    | 19.32 (82.38) |
| SIMPLES2S | ✓  | ✓    | 21.30 (81.22) |
| SIMPLES2S | ✓  | ✓    | 20.95 (81.94) |
| SIMPLES2S | ✓  | ✓    | 28.86 (80.53) |
| SIMPLES2S | ✓  | ✓    | 21.12 (75.63) |
| SIMPLES2S | ✓  | ✓    | 31.17 (81.51) |
| SIMPLES2S | ✓  | ✓    | 19.32 (82.38) |
| SIMPLES2S | ✓  | ✓    | 20.26 (81.54) |
| SIMPLES2S | ✓  | ✓    | 20.14 (83.46) |
| SIMPLES2S | ✓  | ✓    | 21.46 (81.93) |
| SIMPLES2S | ✓  | ✓    | 23.76 (81.68) |
| SIMPLES2S | ✓  | ✓    | 20.68 (83.18) |
| SIMPLES2S | ✓  | ✓    | 22.23 (81.71) |
| SIMPLES2S | ✓  | ✓    | 22.62 (81.15) |
| SIMPLES2S | ✓  | ✓    | 27.66 (81.35) |
| SIMPLES2S | ✓  | ✓    | 24.11 (79.92) |
| SIMPLES2S | ✓  | ✓    | 27.81 (84.67) |

8.1 Qualitative Analysis

Figure 2 shows the attention matrices from our best Copy model (Copy.Yes.RetroExtFixed) and our best SimpleS2S model (SimpleS2S.Yes.RetroFixed) respectively for the same input test sentence. Without an explicit Copy component, the SimpleS2S model cannot predict the words saint and francis, and drifts off after predicting incorrect word flute.

All neural outputs are lowercase due to our preprocessing. Although this slightly affects BLEU, it helps prevent token occurrences getting split due to capitalization.
Figure 2: Attention matrices from a Copy (top) and a simple S2S (bottom) model respectively on the input sentence “Holy Saint Francis, this is a drastic change!” . <s> and </s> are start and stop characters. Darker cells are higher-valued.

semble the ground truth more closely compared to SimpleS2S and Stat . In some cases, it faces issues with repetition (Examples 5 and 7) and fluency (Example 9).

9 Related Work

There have been some prior work on style adaptation. Xu et al. (2012) use phrase table based statistical machine translation to transform text to target style. On the other hand our method is an end-to-end trainable neural network. Saha Roy et al (2015) leverage different language models based on geolocation and occupation to align a text to specific style. However, their work is limited to addition of adjectives and adverbs. Our method can handle more generic transformations including addition and deletion of words.

Pointer networks (Vinyals et al., 2015) allow the use of input-side words directly as output in a neural S2S model, and have been used for tasks like extractive summarization (See et al., 2017) (Zeng et al., 2016) and question answering (Wang and Jiang, 2016). However, pointer networks cannot generate words not present in the input. A mixture model of recurrent neural network and pointer network has been shown to achieve good performance on language modeling task (Merity et al., 2016).

S2S neural models, first proposed by Sutskever et al. (2014), and enhanced with a attention mechanism by Bahdanau et al. (2014), have yielded state-of-the-art results for machine translation (MT), summarization (Rush et al., 2015), etc. In the context of MT, various settings such as multi-source MT (Zoph and Knight, 2016) and MT with external information (Sennrich et al., 2016) have been explored. Distinct from all of these, our work attempts to solve a modern → Shakespearean style English style transformation task. Although closely related to both paraphrasing and MT, our task has some differentiating characteristics such as considerable source-target overlap in vocabulary and grammar (unlike MT), and different source and target language (unlike paraphrasing). Gangal et al. (2017) have proposed a neural sequence-to-sequence solution for generating portmanteau from two English words. Though their task also involves large overlap in target and input, they do not employ any special copying mechanism. Unlike text simplification and summarization, our task does not involve shortening content length.

10 Conclusion

In this paper we have proposed to use a mixture model of pointer network and LSTM to transform Modern English text to Shakespearean style English. We demonstrate the effectiveness of our proposed approaches over the baselines. Our experiments reveal the utility of incorporating input-copying mechanism, and using dictionary constraints for problems with shared (but non-identical) source-target sides and sparse parallel data.

We have demonstrated the transformation to Shakespearean style English only. Methods have to be explored to achieve other stylistic variations corresponding to formality and politeness of text, usage of fancier words and expressions, etc. We release our code publicly to foster further research on stylistic transformations on text. 11

11 https://github.com/harsh19/Shakespearizing-Modern-English
References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. arXiv:1409.0473.

Roger Brown, Albert Gilman, et al. 1960. The pronouns of power and solidarity. Blah.

David L. Chen and William B Dolan. 2011. Collecting highly parallel data for paraphrase evaluation. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1. Association for Computational Linguistics, pages 190–200.

Manaal Faruqui, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2014. Retrofitting word vectors to semantic lexicons. arXiv preprint arXiv:1411.4166.

Varun Gangal, Harsh Jhamtani, Graham Neubig, Eduard Hovy, and Eric Nyberg. 2017. Charmanteau: Character embedding models for portmanteau creation. In Conference on Empirical Methods in Natural Language Processing (EMNLP). Copenhagen, Denmark.

Gaurush Hiranandani, Pranav Maneriker, and Harsh Jhamtani. 2017. Generating appealing brand names. arXiv preprint arXiv:1706.09335.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Eliza Kitis. 1997. Adspar: of our lives: linguistic awareness of powerful advertising. Word & Image 13(3):304–313.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions. Association for Computational Linguistics, pages 177–180.

Pranava Swaroop Madhyastha, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Mapping unseen words to task-trained embedding spaces. arXiv preprint arXiv:1510.02387.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. Computational linguistics 19(2):313–330.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer Sentinel Mixture Models. arXiv preprint arXiv:1609.07843.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1. Association for Computational Linguistics, pages 160–167.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In Proceedings of the 40th annual meeting on association for computational linguistics. Association for Computational Linguistics, pages 311–318.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global Vectors for Word Representation. In EMNLP. volume 14, pages 1532–1543.

Paul Rayson, Dawn Archer, Alistair Baron, Jonathan Culpeper, and Nicholas Smith. 2007. Tagging the Bard: Evaluating the accuracy of a modern POS tagger on Early Modern English corpora. Blah.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. arXiv preprint arXiv:1509.00685.

Rishiraj Saha Roy, Aishwarya Padmakumar, Gunu Prasad Jeganathan, and Ponnurangam Kumaraguru. 2015. Automated Linguistic Personalization of Targeted Marketing Messages Mining User-Generated Text on Social Media. In 16th International Conference on Intelligent Text Processing and Computational Linguistics 2015 (CICLing ’15). Springer International Publishing, pages 203–224.

H Andrew Schwartz, Johannes C Eichstaedt, Margaret L Kern, Lukasz Dziurzynski, Stephanie M Ramones, Megha Agrawal, Achal Shah, Michal Kosinski, David Stillwell, Martin EP Seligman, et al. 2013. Personality, gender, and age in the language of social media: The open-vocabulary approach. PloS one 8(9):e73791.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get To The Point: Summarization with Pointer-Generator Networks. arXiv preprint arXiv:1704.04368.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Controlling politeness in neural machine translation via side constraints. In Proceedings of NAACL-HLT. pages 35–40.

Balaji Vasan Srinivasan, Rishiraj Saha Roy, Harsh Jhamtani, Natwar Modani, and Niyati Chhaya. 2017. Corpus-based automatic text expansion. In CICLING.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Neural information processing systems. pages 3104–3112.
Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In Advances in Neural Information Processing Systems. pages 2692–2700.

Shuohang Wang and Jing Jiang. 2016. Machine comprehension using match-lstm and answer pointer. arXiv preprint arXiv:1608.07905.

Wei Xu. 2014. Data-driven approaches for paraphrasing across language variations. Ph.D. thesis, New York University.

Wei Xu, Alan Ritter, William B Dolan, Ralph Grishman, and Colin Cherry. 2012. Paraphrasing for style. In 24th International Conference on Computational Linguistics, COLING 2012.

Wenyuan Zeng, Wenjie Luo, Sanja Fidler, and Raquel Urtasun. 2016. Efficient Summarization with Read-Again and Copy Mechanism. arXiv preprint arXiv:1611.03382.

Barret Zoph and Kevin Knight. 2016. Multi-source neural translation. arXiv preprint arXiv:1601.00710.