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

Harsh Jhamtani [*], **Varun Gangal** [*], **Eduard Hovy, Eric Nyberg**
Language Technologies 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 largely 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 over the strongest baseline. We publicly release our code to foster further research in this area. [1]

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 effectively adapt text to different styles. However, manually transforming text to a 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

No	Type	Text
1	MODERN	Oh my, my bones ache so much
	ORIGINAL	Fie, how my bones ache !
	COPY	fie, how my bones ache !
	SIMPLES2S	you'll be, sir, what the bones are tired .
	STAT	Oh my, my bones ache so much .
2	MODERN	I am in a rush .
	ORIGINAL	I stand on sudden haste .
	COPY	i stand on sudden haste .
	SIMPLES2S	i'm stand right here .
	STAT	I am in a Fly
3	MODERN	Give my compliments to your lady
	ORIGINAL	Commend me to thy lady
	COPY	commend me to your lady
	SIMPLES2S	give my regards to your lady
	STAT	give my praises to your lady
4	MODERN	Showing mercy by pardoning killers only causes more murders .
	ORIGINAL	Mercy but murders, pardoning those that kill .
	COPY	mercy but murders, those those who kill us .
	SIMPLES2S	but except the murders to those murders to kill you .
	STAT	of mercy by pardoning killers causes more dire.
5	MODERN	Holy Saint Francis, this is a drastic change !
	ORIGINAL	Holy Saint Francis, what a change is here !
	COPY	holy saint francis, what a change is here !
	SIMPLES2S	it's the holy flute, what's the changed !
	STAT	Holy Saint Francis, this is a drastic change !
6	MODERN	was that my father who left here in such a hurry ?
	ORIGINAL	Was that my father that went hence so fast ?
	COPY	was that my father that went went so fast ?
	SIMPLES2S	was that my father was so that ?
	STAT	was that my father that left here in such a haste ?
7	MODERN	Give me one kiss and I'll go down .
	ORIGINAL	One kiss, and I'll descend .
	COPY	one kiss me, and I'll descend .
	SIMPLES2S	one kiss,and I come down .
	STAT	Give me a kiss, and I'll go down .
8	MODERN	then the window lets day in, and life goes out the window .
	ORIGINAL	Then, window, let day in and life out .
	COPY	then, window out, and day life .
	SIMPLES2S	then she is just a life of life, let me life out of 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).

[*] denotes equal contribution

[1] https://github.com/harsh19/Shakespearizing-Modern-English

(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 different styles. We consider the problem of transforming text written in modern English text to Shakepearean 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* [2] 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 dramatic screenplay whereas the *parallel* modern English sentences are meant to be simplified explanation for high-school students.

Prior works in this field leverage a language model for the 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 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.

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 leverage a dictionary providing mapping between Shakespearean words and modern English words to retrofit pre-trained word embeddings. Incorporating this extra information enables our model to perform well in spite of small size 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).

	Original	*Modern*
# Word Tokens	217K	200K
# Word Types	12.39K	10.05K
Average Sentence Length	11.81	10.91
Entropy (Type.Dist)	6.15	6.06
∩ Word Types	6.33K	

Table 2: Dataset Statistics

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*[3]. This dataset was compiled by Xu et al. (2014; 2012) and is freely available on github.[4] 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

[2] https://en.wikipedia.org/wiki/Early_Modern_English

[3] www.sparknotes.com

[4] http://tinyurl.com/ycdd3v6h

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 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 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 amd 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 $\mathbf{x}, \mathbf{y}$ be the some input - output sentence pair in the dataset. Both input $\mathbf{x}$ as well as output $\mathbf{y}$ are sequence of tokens. $\mathbf{x} = \mathbf{x}_1\mathbf{x}_2...\mathbf{x}_{T_{enc}}$, where T_{enc} represents the length of the input sequence $\mathbf{x}$. Similarly, $\mathbf{y} = \mathbf{y}_1\mathbf{y}_2...\mathbf{y}_{T_{dec}}$. Each of $\mathbf{x}_i$, $\mathbf{y}_j$ is 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 Shakepearean vocabularies i.e. $V = V_{shakespeare} \cup V_{modern}$. 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 $\mathbf{x}$, $E_{enc}(\mathbf{x})$ is given as $(E_{enc}(\mathbf{x}_1), E_{enc}(\mathbf{x}_2), ...)$.

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

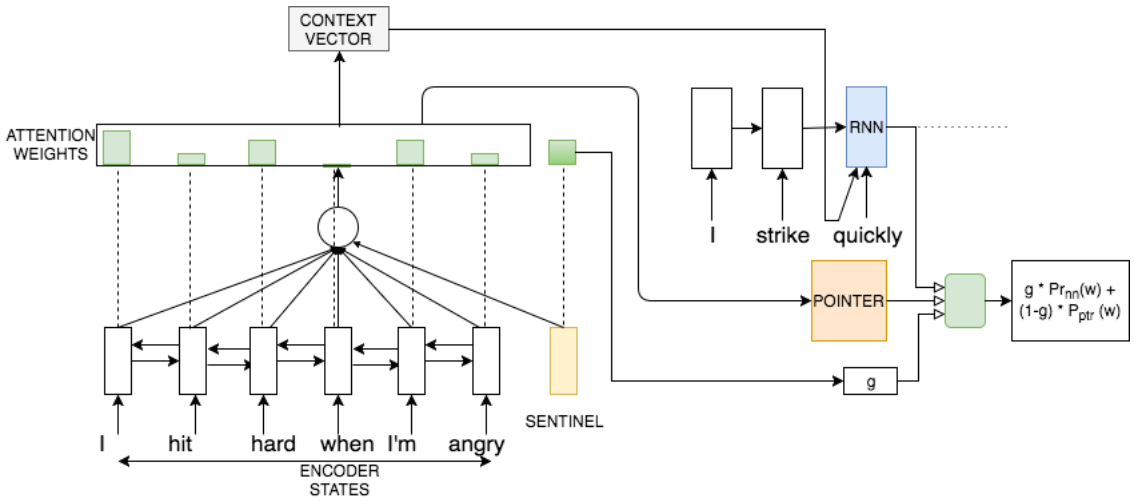

Figure 1: Depiction of our overall architecture (showing decoder step 3). Attention weights are computed using previous decoder hidden state h_2, encoder representations, and sentinel vector. Attention weights are shared by decoder RNN and pointer models. The final probability distribution over vocabulary comes from both the decoder RNN and the pointer network. Similar formulation is used over all decoder steps

new set of embeddings $q_i \in Q$ to minimize the following objective:

$$f(Q) = \delta \sum_{i=1}^{i=|V|} (p_i - q_i)^2 + \omega \sum_{(i,j)\in C} (q_i - q_j)^2 \quad (1)$$

We use their off-the-shelf implementation [5] to encode the dictionary constraints into our pretrained embeddings, setting $C = L$ and using suggested default hyperparameters for δ, ω 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] github.com/mfaruqui/retrofitting

5 Method Description

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

5.1 Encoder model

Let $\overrightarrow{LSTM_{enc}}$ and $\overleftarrow{LSTM_{enc}}$ represent the forward and reverse encoder. $\mathbf{h}_t^{\overrightarrow{enc}}$ represent hidden state of encoder model at step t ($\mathbf{h}_t^{\overrightarrow{enc}} \in \mathbb{R}^H$). The following equations describe the model:

$$\mathbf{h}_0^{\overrightarrow{enc}} = \overrightarrow{0}, \mathbf{h}_{|x|}^{\overleftarrow{enc}} = \overrightarrow{0} \quad (2)$$

$$\mathbf{h}_t^{\overrightarrow{enc}} = \overrightarrow{LSTM_{enc}}(\mathbf{h}_{t-1}^{enc}, E_{enc}(\mathbf{x}_t)) \quad (3)$$

$$\mathbf{h}_t^{\overleftarrow{enc}} = \overleftarrow{LSTM_{enc}}(\mathbf{h}_{t+1}^{enc}, E_{enc}(x_t)) \quad (4)$$

$$\mathbf{h}_t^{enc} = \mathbf{h}_t^{\overrightarrow{enc}} + \mathbf{h}_t^{\overleftarrow{enc}} \quad (5)$$

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 $\mathbf{h}_t^{dec}$ represent the hidden state of the decoder LSTM at step t. Let $E_{dec}(\mathbf{y}_{t-1})$ represent the decoder side embeddings of previous step output. We use special $START$ symbol at $t = 1$.

We first compute a query vector, which is a linear transformation of $\mathbf{h}_{t-1}^{dec}$. A sentinel vector $\mathbf{s} \in \mathbb{R}^H$ is concatenated with the encoder states to create $F_{att} \in \mathbb{R}^{(T_{enc}+1)\times H}$, where T_{enc} represents the number of tokens in encoder input sequence $\mathbf{x}$. A normalized attention weight vector $\boldsymbol{\alpha}^{norm}$ 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 probabilties.

$$\mathbf{q} = \mathbf{h}_{t-1}^{dec}\, W_q \qquad W_q \in \mathbb{R}^{H\times H} \quad (6)$$

$$F_{att} = concat(\mathbf{h}_{1..T_{enc}}^{enc}, \mathbf{s}) \qquad F_{att} \in \mathbb{R}^{(T_{enc}+1)\times H} \quad (7)$$

$$\boldsymbol{\alpha}_i = \sum_{j=1}^{H}(tanh(F_{att}^{(ij)}\, \mathbf{q}_j)) + \mathbf{b}_i \qquad \boldsymbol{\alpha}_i, \mathbf{b}_i \in \mathbb{R} \quad (8)$$

$$\boldsymbol{\alpha}^{norm} = softmax(\boldsymbol{\alpha}) \qquad \boldsymbol{\alpha}^{norm} \in \mathbb{R}^{T_{enc}+1} \quad (9)$$

$$\boldsymbol{\beta} = \boldsymbol{\alpha}^{norm}_{1,2,\ldots,T_{enc}} \qquad \boldsymbol{\beta} \in \mathbb{R}^{T_{enc}} \quad (10)$$

$$g = \boldsymbol{\alpha}^{norm}_{T_{enc}+1} \qquad g \in \mathbb{R} \quad (11)$$

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:

$$P_t^{PTR}(w) = \sum_{\mathbf{x}_j = w} (\boldsymbol{\beta}_j) \quad (12)$$

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:

$$\mathbf{c}_t = \sum_{i=1}^{T_{enc}} \boldsymbol{\beta}_i\, \mathbf{h}_i^{enc} \quad (13)$$

$$\mathbf{h}_t^{dec} = LSTM(\mathbf{h}_{t-1}^{dec}, [\text{concat}(E_{dec}(\mathbf{y}_{t-1}), \mathbf{c}_t)]) \quad (14)$$

$$P_t^{LSTM} = \text{softmax}(W_{out}[\text{concat}(\mathbf{h}_t^{dec}, \mathbf{c}_t)] + \mathbf{b}^{out}) \quad (15)$$

During training, we feed the ground truth for $\mathbf{y}_{t-1}$, 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:

$$P_t(w) = g \times P_t^{LSTM}(w) + (1-g) \times P_t^{PTR}(w) \quad (16)$$

$P_t^{PTR}(w)$ takes a non-zero value only if w occurs in input sequence, otherwise it is 0. Forcing $g = 0$ would correspond to not having a *Copy* component, reducing the model to a plain attentional S2S model, which we refer to as a *SimpleS2S* model.

6 Loss functions

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

$$-\sum_{t=1}^{T_{dec}} \log p\left(P_t\left(\mathbf{y}_t\right)\right) \quad (17)$$

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:

$$-\sum_{t=1}^{T_{dec}} \log(g^{(t)} + \sum_{x_j = y_t} (\beta_j^{(t)})) \quad (18)$$

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 æwhich 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 admittable 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 [6].

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.

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[7] 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

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

where $Ngram(x,n)$ denotes the set of n-grams of length n in sentence x, and N is the maximum length of ngram 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 [8]. 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[9]. 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 [10]. 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.

[6] thou-thyself and you-yourself

[7] http://tinyurl.com/yben45gm

[8] This is expected given the small parallel data

[9] Empirically, we observed that beam search does not give improvements for our task

[10] Since effect is small and uniform, we report BLEU before post-processing in Table 3

- **Copy component**: We can observe from Table 3 that the various *Copy* models each outperform their *SimpleS2S* counterparts by atleast 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 atleast 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* get 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.

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*.

Model	Sh	Init	BLEU (PINC)
AS-IT-IS	-	-	21.13 (0.0)
DICTIONARY	-	-	17.00 (26.64)
STAT	-	-	**24.39** (32.30)
SIMPLES2S	×	$NoneVar$	11.66 (85.61)
	×	$PlainVar$	9.27 (86.52)
	×	$PlainExtVar$	8.73 (87.17)
	×	$RetroVar$	10.57 (85.06)
	×	$RetroExtVar$	10.26 (83.83)
	✓	$NoneVar$	11.17 (84.91)
	✓	$PlainVar$	8.78 (85.57)
	✓	$PlainFixed$	8.73 (89.19)
	✓	$PlainExtVar$	8.59 (86.04)
	✓	$PlainExtFixed$	8.59 (89.16)
	✓	$RetroVar$	10.86 (85.58)
	✓	$RetroFixed$	11.36 (85.07)
	✓	$RetroExtVar$	11.25 (83.56)
	✓	$RetroExtFixed$	**10.86** (88.80)
COPY	×	$NoneVar$	18.44 (83.68)
	×	$PlainVar$	20.26 (81.54)
	×	$PlainExtVar$	20.20 (83.38)
	×	$RetroVar$	21.25 (81.18)
	×	$RetroExtVar$	21.57 (82.89)
	✓	$NoneVar$	22.70 (81.51)
	✓	$PlainVar$	19.27 (83.87)
	✓	$PlainFixed$	21.20 (81.61)
	✓	$PlainExtVar$	20.76 (83.17)
	✓	$PlainExtFixed$	19.32 (82.38)
	✓	$RetroVar$	22.71 (81.12)
	✓	$RetroFixed$	**28.86** (80.53)
	✓	$RetroExtVar$	20.95 (81.94)
	✓	$RetroExtFixed$	**31.12** (79.63)
COPY+SL	×	$NoneVar$	17.88 (83.70)
	×	$PlainVar$	20.22 (81.52)
	×	$PlainExtVar$	20.14 (83.46)
	×	$RetroVar$	21.30 (81.22)
	×	$RetroExtVar$	21.52 (82.86)
	✓	$NoneVar$	22.72 (81.41)
	✓	$PlainVar$	21.46 (81.39)
	✓	$PlainFixed$	23.76 (81.68)
	✓	$PlainExtVar$	20.68 (83.18)
	✓	$PlainExtFixed$	22.23 (81.71)
	✓	$RetroVar$	22.62 (81.15)
	✓	$RetroFixed$	27.66 (81.35)
	✓	$RetroExtVar$	24.11 (79.92)
	✓	$RetroExtFixed$	27.81 (84.67)

Table 3: Test BLEU results. *Sh* denotes encoder-decoder embedding sharing (*No*=×,*Yes*=✓) . *Init* denotes the manner of initializing embedding vectors. The *-Fixed* or *-Var* suffix indicates whether embeddings are fixed or trainable. COPY and SIMPLES2S denote presence/absence of *Copy* component. +SL denotes sentinel loss.

Table 1 presents model outputs[11] for some test examples. In general, the *Copy* model outputs re-

[11] All neural outputs are lowercase due to our preprocessing. Although this slightly affects BLEU, it helps prevent token occurrences getting split due to capitalization.

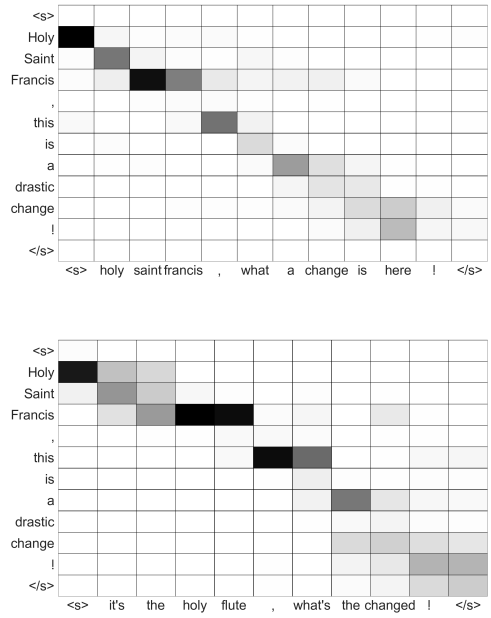

Figure 2: Attention matrices from a *Copy* (top) and a *simple S2S* (bottom) model respectively on the input sentence *"Holy Saint Francis, this is a drastic change!"* . $< s >$ and $< /s >$ are start and stop characters. Darker cells are higher-valued.

semble the ground truth more closely compared to *SimpleS2S* and *Stat* . In some cases, it faces issues with repetition (Examples 4 and 6) and fluency (Example 8).

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 English $\rightarrow$ Shakespearean 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 a portmanteau given two English root-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. [12].

[12]https://github.com/harsh19/Shakespearizing-Modern-English

Acknowledgements

We thank Taylor Berg-Kirkpatrick and anonymous reviewers for their comments. This research was supported in part by DARPA grant FA8750-12-2-0342 funded under the DEFT program.

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. *Article*.

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*, pages 190–200. Association for Computational Linguistics.

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. Adspart 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*, pages 177–180. Association for Computational Linguistics.

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*, pages 160–167. Association for Computational Linguistics.

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*, pages 311–318. Association for Computational Linguistics.

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. *Article*.

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, Guna Prasaad 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)*, pages 203–224. Springer International Publishing.

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*.