”Conversation with Shakespeare” using Deep Learning

Sai Nikhil Maram

Abstract: This project aims to build a bot that will allow the user to converse in modern English and get the corresponding response in ”Shakespearean” or Elizabethan English. It uses sequence2sequence models to build this bot. Two models are employed, one after the other to the user's input, to give an output in Elizabethan English. The first model generates the output to the user's input in modern English, and the second model converts the generated output to Elizabethan English. Our results show that the bot is able to respond to modern English inputs with a reasonable quality of response. The outputs can only be evaluated qualitatively by human judgment as of now, due to the lack of an existing modern English-”Shakespearean” English conversations dataset.

1. Introduction

Most of us have studied Shakespeare and his work in high school and have wondered what it would have been like to have lived in that time, and what it would be like to speak to someone in that time, in their style of English. Our project attempts to give a little flavor of Shakespearean times by giving the user the opportunity to essentially ”converse with Shakespeare”. To the best of our knowledge, such a project has not been done before.

2. Objective

Our aim was to build a bot that will allow the user to have a conversation with Shakespeare. The bot replies to modern English conversation in English similar to that spoken in Shakespeare's time and in his plays. The bot has not been provided with information specific to Shakespeare's life and therefore simply emulates any individual from Elizabethan times in terms of style of language.

3. Related Work

In recent years, we have seen many papers on style transfer, which focus on translating from one language to another. A very recent paper on style transfer from modern English to ”Shakespearean” English (Jhamtani et al., 2017) was one of the first we could find on style transfer within the English language. We also came across the work of (Shen et al., 2017) on style transfer with non-parallel text corpora. Our project however, uses parallel text corpora for the purpose of building the style transfer model.
In the field of conversational bots we have seen the work of (Li et al., 2017) and (Serban et al., 2016) where dialogue systems were built using Neural Network models.

4. Overview of the Model

The System consists of two parts

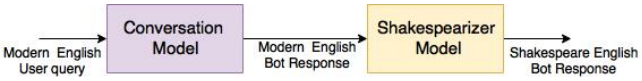

Figure 1. System Architecture

1. Conversation Model : This dialogue model takes an input from the user in modern English and generates the output response pertaining to the input in modern English. This model has been further discussed in Section 5.
2. Shakespearizer Model : The Modern English output obtained from the Conversation Model is fed here and translated to Shakespearean English. This model has been further discussed in Section 6.

5. Conversation Model Architecture

The conversation model is a sequence2sequence model, containing an Encoder and a Decoder. The encoder encodes the user dialogue into the context space. The decoder generates a response to the user's input by decoding the context space. This model has been depicted in Figure 2.

5.1. Encoder

A multi-layer RNN (2 layers) is used to capture the context of the user's input. Each RNN is an LSTM with 512 memory units. Let h_t^{enc} be the hidden state of the encoder at time t. Let $LSTM_{enc}$ be the 2-layer LSTM unit.

$$h_0^{enc} = 0 \tag{1}$$

$$h_t^{enc} = LSTM_{enc}(h_{t-1}^{enc}, E_{enc}^{conv}(x_t)) \tag{2}$$

E_{enc}^{conv} is the embedding matrix for the encoder.

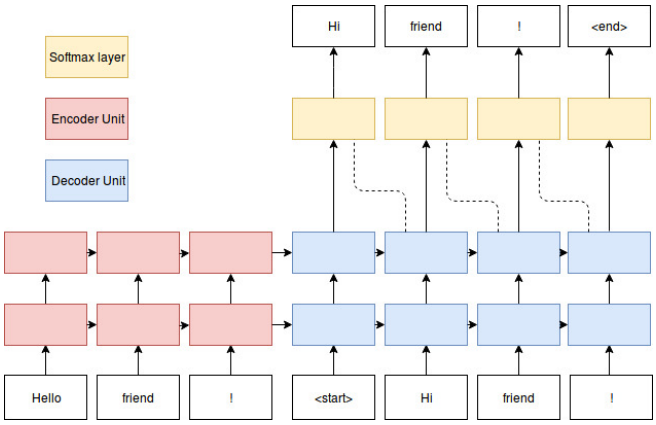

Figure 2. Encoder-Decoder model

5.2. Decoder

A multi-layer RNN (2 layers) is used to capture the context of the user's input. Each RNN is an LSTM with 512 memory units. Let h_t^{dec} be the hidden state of the decoder at time t. Let $LSTM_{dec}$ be the 2-layer LSTM unit. The initial hidden state of $LSTM_{dec}$ is the final hidden state of the encoder.

$$h_0^{dec} = h_{T_{enc}}^{enc} \tag{3}$$

$$h_t^{dec} = LSTM_{dec}(h_{t-1}^{enc}, E_{dec}^{conv}(y_{t-1})) \tag{4}$$

$$y_t = softmax(W_o^{conv} h_t^{dec} + b_o^{conv}) \tag{5}$$

y_t is the output of the decoder at time t. E_{dec}^{conv} is the embedding matrix for the decoder. We have used the same embedding matrix for the encoder and the decoder.

5.3. Loss Function

The cross entropy loss function is used to train the model.

$$L(\hat{y}, y) = -\sum_{y \epsilon y^{dec}} \hat{y} \log(y) \tag{6}$$

Here $\hat{y}$ is the actual output and y is the predicted output of the decoder module.

6. "Shakespearizer" Model Architecture

We have used an attention based sequence2sequence model for style transfer (translating from modern English to Shakespearean English). The Encoder is a bi-directional LSTM that captures the context of the input sentence. The decoder model is a mixture of two components, namely the RNN module and the Pointer Network module. The RNN module is used to predict the next word, given all words in the vocabulary. The Pointer network is used to predict the next word, given the words in the input. This is particularly useful because both Modern and Shakespearean English have a lot of words in common. The output of each decoder module undergoes a weighted addition to determine the final output. This model has been depicted in Figure 3.

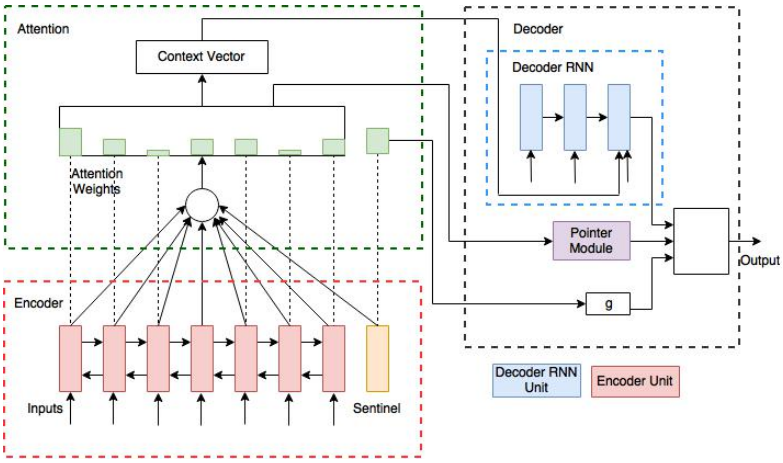

Figure 3. Shakespearizer Model

6.1. Encoder

A Bi-directional LSTM is used to capture the context of the modern English input. Let $h_t^{\overrightarrow{enc}}$ & $h_t^{\overleftarrow{enc}}$ represent the hidden state of the forward and backward LSTMs respectively at time stamp t, and let $\overrightarrow{LSTM_{enc}}$ & $\overleftarrow{LSTM_{enc}}$ be the forward and backward LSTMs respectively.

$$h_0^{\overrightarrow{enc}} = 0, h_{|x|}^{\overleftarrow{enc}} = 0 \tag{7}$$

$$h_t^{\overrightarrow{enc}} = \overrightarrow{LSTM_{enc}}(h_{t-1}^{\overrightarrow{enc}}, E_{enc}(x_t)) \tag{8}$$

$$h_t^{\overleftarrow{enc}} = \overleftarrow{LSTM_{enc}}(h_{t+1}^{\overleftarrow{enc}}, E_{enc}(x_t)) \tag{9}$$

$$h_t^{enc} = h_t^{\overrightarrow{enc}} + h_t^{\overleftarrow{enc}}; h_t^{enc} \epsilon R^H \tag{10}$$

Here E_{enc} represents the embedding matrix of the encoder. The hidden state at each time stamp is a sum of the forward and backward hidden states at each step rather than a concatenation of the two because of the presence of a relatively small amount of data.

6.2. Attention

Let h_t^{dec} represent the hidden state of a decoder at time t, and E_{dec} represent the word embeddings of the decoder. At each time stamp t, the previous hidden state of the decoder and all hidden states of the encoder along with a sentinel vector(s) are taken into consideration while generating the attention vector $\mathcal{F}_{att}$. The normalized attention vector is used in determining the attention weights given to

the pointer module and RNN module (g) in the decoder.

$$q = h_{t-1}^{dec} W_q \tag{11}$$

$$\mathcal{F}_{att} = concat(h_{1\cdots T_{enc}}^{enc}, s) \tag{12}$$

$$\alpha_i = \sum_{j=1}^{H} (tanh(\mathcal{F}_{att}^{(ij)}, q_j)) + b_i \tag{13}$$

$$\alpha^{norm} = softmax(\alpha) \tag{14}$$

$$\beta = \alpha_{1,2\cdots T_{enc}}^{norm} \tag{15}$$

$$g = \alpha_{T_{enc+1}}^{norm} \tag{16}$$

6.3. Pointer Module

The Pointer Module is used in determining the next word, given the words in the input. Usually in Machine Translation, the two languages over which translation is being carried out do not have many words in common. However in our case, Modern and Shakespearean English have many common words. For this reason, we employ the pointer module. It gives us location-based attention, which tells us the probability of a word in the input being the next word of the decoder at time t.

$$P_t^{PTR} = \sum_{x_j = w} \beta_j \tag{17}$$

x_j corresponds to the word at index j and β_j corresponds to the attention value at index j. $P_t^{PTR}(w)$ gives the probability of w being the next word when chosen by the Pointer Module at time t.

6.4. RNN module

In the RNN module, the hidden state at a time t depends on the previous hidden state, previous output, the context vector generated from the hidden states of the decoder and their corresponding attention weights.

$$c_t = \sum_{i=1}^{T_{enc}} \beta_i h_i^{enc} \tag{18}$$

$$h_t^{dec} = LSTM(h_{t-1}^{dec}, [concat(E_{dec}(y_{t-1}), c_t])) \tag{19}$$

$$P_t^{RNN} = softmax(W_o([concat(h_{dec}, c_t))] + b_o) \tag{20}$$

$P_t^{RNN}(w)$ gives the probability of w being the next word when chosen by the RNN Module at time t. y_{t-1} is the ground truth fed during training and is the output of the previous step during testing.

6.5. Output

The attention weight given to the sentinel vector decides the probability of picking a word from the RNN module. P_t^{RNN} in turn will have the probability of each word in the vocabulary. So the probabilities of the next words can be calculated as shown in equation 21.

$$P_t(w) = g \times P_t^{RNN}(w) + (1 - g) \times P_t^{PTR}(w) \tag{21}$$

Hence the probability of the next word appearing in the output is the weighted average of probability of the words in the Pointer and RNN modules. The weights are calculated from the previous hidden state of the decoder and all the hidden states of the encoder.

6.6. Loss Function

Cross entropy is used to train this model.

$$L(\hat{y}, y) = -\sum_{y \epsilon y^{dec}} \hat{y} \log(y) \tag{22}$$

Here $\hat{y}$ is the actual output and y is the predicted output by the decoder module.

7. Experiments

In this section, we will discuss the datasets used and the preprocessing and training details of our models.

7.1. Datasets

The datasets we used to build our bot are as follows:

1. Cornell Movie Dialog Corpus (for the Conversation model) This contains dialogs from 617 movies with a total of 220k conversations.
2. Sixteen of Shakespeare's plays and their modern English translations by Sparknotes (for the "Shakespearizer" model) This contains 12.3k unique Shakespearean words, 10k modern English words and 6.3k intersection words.

7.2. Preprocessing

7.2.1. Conversation Model

The RNN unit is run for a specified input length. All inputs are either truncated or padded to fit that specified length. All inputs to the Encoder are left padded and inputs to the decoder are right padded. During training, the decoder input is appended with a $< start >$ token in the beginning and an $< end >$ token at the end. The words in the dialogue are then converted to indexes based on the vocabulary obtained from the pre-trained word embeddings. Thus, the input dialogue is converted to an array of indexes corresponding to each word.

7.2.2. Shakespearizer Model

We do not use pre-trained embeddings for the training of this model, due to their bias towards modern English, hav-

ing been trained on the same. Hence the vocabulary is built from the words present in Shakespeare's plays and the Penn Treebank data set. Embeddings are created using Word2vec on this vocabulary. A retro-fitting mechanism has also been adapted for improving the embeddings, using existing Shakespearean dictionaries. Retro-fitting ensures that Shakespearean words and their modern English translations' embedding vectors are close to each other.

7.3. Training

7.3.1. Conversation Model

We used an RNN sequence2sequence legacy tensorflow model to generate sentence predictions. It has a multi-layered (2 layers) RNN structure. Each RNN is an LSTM with a memory unit of 512. The dynamic rolling of each LSTM cell has been considered. During the training phase, we used a batch size of 256, and trained the model over 30 epochs (supervised training). Pre-trained Glove Embeddings were used with an embedding dimension of 300. We used an Adam optimizer with an initial learning rate of 0.02, and cross entropy as our loss function. A dropout mechanism with a probability of 0.9 was used, so as to avoid over-fitting of the data. We used perplexity as a metric for evaluation. The parameters trained during this phase were the LSTM weights. While training, the input to the decoder RNN module is fed by the user while during evaluation the output generated by the previous LSTM unit is fed as an input to the current LSTM unit. Our training dataset comprised of 80% of the dialogues, and the remaining 20% were used for validation.

7.3.2. Shakespearizer Model

A Bi-directional LSTM was used to build the Encoder while an uni-directional LSTM was used to build the decoder. Each LSTM's size is 192 units. Encoder inputs are left padded and decoder inputs are right padded. Words are then embedded into a 192 dimensional vector. Cross Entropy was used as the loss function. For optimization, we used the Adam Optimizer with an initial learning rate of 0.001. The batch size we used while training this model was 192, and training was carried out for 15 epochs. We used BLEU score to detect the point for early stop. Fourteen of Shakespeare's plays were used to train the model, and two were used for validation.

8. Results

Each module is trained independently due to the lack of availability of a single data set that would allow us to train the entire model end-to-end. In order to generate the final Shakespearean dialogue, both pre-trained models have been combined.

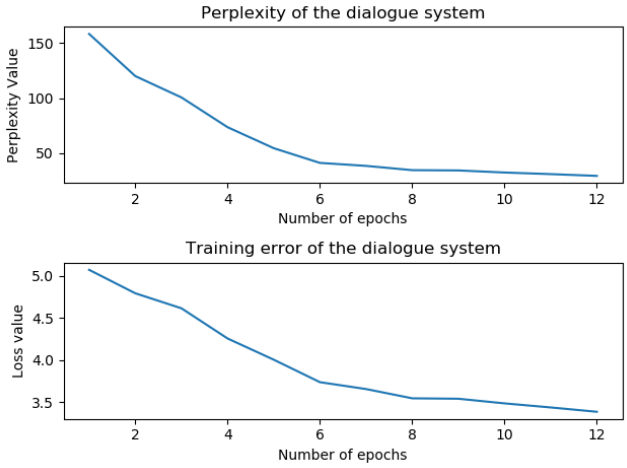

Figure 4. Perplexity and Training Error of the Dialog System

8.1. Conversation Model

Here are some conversations we had with our bot (phase 1) where it responds in modern English to a modern English input.

Q: Would you like some ice-cream?
A: No.

Q: When are the fireworks?
A: Tomorrow morning.

Q: What is your name?
A: Rufus.

Q: Are you okay?
A: I'm fine.

Q: Would you like to dance?
A: Sure.

The conversations are quite short (simple question and answer), and the response to a question remains the same when asked multiple times. The length of the input to the conversation model has been limited to 10 tokens.

8.2. "Shakespearizer" Model

The "Shakespearizer" model, after training on the dataset, gives us a BLEU score of 30.32. Given below are some sample inputs and corresponding outputs to this model.

Input: Good morning, father.
Output: Good morrow, father.

Input: Stay here for a moment.
Output: Stay but a little.

Input: Are you trying to start a fight?
Output: Do you quarrel sir?

Input: So what?
Output: What of this?

8.3. Final Model

Here are some sample conversations with our final (combined) model.

I: When are the fireworks?
O: Tomorrow, then, tomorrow.

I: Let me see what I can do.
O: I will be back, back in this.

I: Im not sure.
O: Thou liest not a fool, fool.

I: Maybe if you tell me things?
O: Thou liest not.

I: Are you alright?
O: Im not gamesome.

I: Would you like to dance?
O: And I have done.

9. Discussion

Due to the unavailability of a dataset that would allow us to check the accuracy of the responses generated by our final model, we must rely on our knowledge of Elizabethan English to qualitatively validate the results. Until such a dataset is created, or we are able to find an effective evaluation metric for the same, we will be unable to provide quantitative evaluations for our final results.

10. Conclusion

Our Conversation with Shakespeare bot effectively has short conversations with users, replying in "Shakespearean" or Elizabethan English. Our two step model first computes the modern English response to the user's input. The next step of the model then translates this modern English response to Elizabethan English, and returns this response to the user. We used an RNN sequence2sequence model for the first step of the model, and an attention-based sequence2sequence model for the second step. Our final model gives promising results for short conversations with "Shakespeare" and is certainly a delight to talk to.

References

Fu, Zhenxin, Tan, Xiaoye, Peng, Nanyun, Zhao, Dongyan, and Yan, Rui. Style transfer in text: Exploration and evaluation. *arXiv preprint arXiv:1711.06861*, 2017.

Jhamtani, Harsh, Gangal, Varun, Hovy, Eduard, and Nyberg, Eric. Shakespearizing modern language using copy-enriched sequence-to-sequence models. *arXiv preprint arXiv:1707.01161*, 2017.

Li, Xuijun, Chen, Yun-Nung, Li, Lihong, and Gao, Jianfeng. End-to-end task-completion neural dialogue systems. *arXiv preprint arXiv:1703.01008*, 2017.

Serban, Iulian Vlad, Sordoni, Alessandro, Bengio, Yoshua, Courville, Aaron C, and Pineau, Joelle. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*, volume 16, pp. 3776–3784, 2016.

Shen, Tianxiao, Lei, Tao, Barzilay, Regina, and Jaakkola, Tommi. Style transfer from non-parallel text by cross-alignment. pp. 6833–6844, 2017.

Xu, Wei, Ritter, Alan, Dolan, Bill, Grishman, Ralph, and Cherry, Colin. Paraphrasing for style. *Proceedings of COLING 2012*, pp. 2899–2914, 2012.