DL Project Interim Report: Shakespeare - English Sequence to Sequence Modeling

Morris Kraicer
Johns Hopkins University
mkraice1@jhu.edu

Riley Scott
Johns Hopkins University
rscott39@jhu.edu

William Watson
Johns Hopkins University
billwatson@jhu.edu

Abstract

We present a interim report on our efforts to contruct a sequence-to-sequence neural translation model with attention. We present the inital work done on data procurement and processing, and provide statitics on our datasets. Additionally, we describe in detail our models that we are experimenting with. Finally, we describe the overall structure of our code to support sequence to sequence modeling.

1 Introduction

We seek to apply a suite of Encoder-Decoder models with varying attention mechanisms and teacher forcing rates to a corpus of Modern English - Original Shakespeare data. Discussion on our procurement is in Section 2 and our processing methods in Section 3. We wanted to apply a style transfer between the two forms of writing, and more specifically test in models currently used in decoding languages (Section 4) can be used in this capcity. We will experment with several model combinations mentioned in Figure 3 and in the final report detail our sucesses and failures. Our current progress in detailed in Figure 4. Diagrams for our models can be found in the appendix.

2 Data Procurement

Most of our data was procured from [6]. This includes all of Shakespeares plays translated line by line into modern English. However, since the data was aligned, not all of the original lines from the plays are included (the sentences could not be aligned properly). [1]

[1] `https://github.com/cocoxu/Shakespeare`

File	**Line Count**
train.snt.aligned	18,444
dev.snt.aligned	2,107
test.snt.aligned	528
total	21,079

Figure 1: Data Pair Counts for Shakespeare–English Corpus

The training vocabulary sizes are 11,538 source (original) words and 9,024 target (modern) words.

3 Preprocessing

We describe our data processing algorithms to create test, development, and training samples in a simple and easy format.

3.1 SOS and EOS

We encapsulate every sentence with two special tokens: SOS and EOS. The Start of Sentence token (SOS) signals the start of a sequence, and allows us to map our first real word to one that most likely starts a sentence.

We use the End of Sentence (EOS) token to signal the end of the sequence, and always comes after punctuation.

By incorporating these special tokens, we can signal to the model the start and end of a sequence, and helps training overall.

3.2 Proper Nouns

Proper nouns are unique in both corpus, and have direct translations. In order to reduce vocabulary size and aggregating the learning of all proper nouns, we replace all proper nouns with the following token: propn. Thus our model should learn to map propn to propn, and can utilize the encoding to learn the most likely token following its usage. We use SpaCy's[2] nlp models to identify proper nouns in each sentence, and replace the tokens.

3.3 Miscellaneous

Additionally, we lower case all input. We use the Natural Language ToolKit's (NLTK) tokenization algorithm to split punctation, contrations, etc for each word.

3.4 Train, Validation, Test Split

We randomly shuffle and split the combined preprocessed dataset into three sets: Train, Dev, and Test. We opted for a 87.5/10/2.5 split to reduce the appearnce of unkown tokens (UNK). We provide statistics for the data.

4 Architectures

We seek to develop several models to improve our translations, incorporating context, attention, and novel training methods. It will incorporate an encoder-decoder style model [2] [5].

4.1 Encoders

The encoder (or *inference network*) receives an input token sequence $\vec{x} = [x_1, \ldots, x_n]$ of length n and processes this to create an output encoding. The result is a sequence $\vec{h} = [h_1, \cdots, h_{T_x}]$ that maps to the input sequence $\vec{x}$.

4.1.1 Baseline RNN Encoder

For the baseline encoder, we implemented a simple Embedding + RNN encoder. This accepts an input sequence $\vec{x}$ and encoders the sequence using the lookup embeddings and forward context.

$$h_t = \tanh(W_{ih}x_t + b_{ih} + W_{hh}h_{t-1} + b_{hh}) \tag{1}$$

Howver, this is the simplest model, prone to the vanishing gradient problem on large sequences. In addition, this model has the lowest capacity to learn. Nonetheless, we will present results for our baseline.

[2] `https://spacy.io`

4.1.2 GRU Encoder

An obvious improvement to our encoding scheme would be to replace the RNN layer with a GRU. A GRU encodes a forward sequence using more complex equations to improve capacity and learning.

$$\begin{aligned}
r_t &= \sigma(W_{ir}x_t + b_{ir} + W_{hr}h_{t-1} + b_{hr}) \\
z_t &= \sigma(W_{iz}x_t + b_{iz} + W_{hz}h_{t-1} + b_{hz}) \\
n_t &= \tanh(W_{in}x_t + b_{in} + r_t \circ (W_{hn}h_{t-1} + b_{hn})) \\
h_t &= (1 - z_t) \circ n_t + z_t \circ h_{t-1}
\end{aligned} \tag{2}$$

GRUs help with vanishing gradients, increases our models capacity to learn, and uses less parameters than the LSTM layer. Hence, for our purposes, we used a GRU over the LSTM.

4.1.3 Bidirectional GRU Encoder

An extension to our encoding scheme will consider the full context of words immediately before an after it. This is doen by running a GRU layer on the forward and backward sequence and combining each tensor. Hence we use a bidirectional GRU as laid forth in [1].

$$\begin{aligned}
\overrightarrow{h_f} &= \mathrm{GRU}(\overrightarrow{input}) \\
\overleftarrow{h_b} &= \mathrm{GRU}(\overleftarrow{input}) \\
h_o &= \overrightarrow{h_f} \parallel \overrightarrow{h_b}
\end{aligned} \tag{3}$$

PyTorch concatenates the resulting tensors, doubling the hidden output size of a normal GRU. Other libraries allow for concatenatation, averaging, and summing. We use PyTorch's implementation of a bidirectional GRU.

4.1.4 Implementation

All encoders share the same model architecture, except for the recurrent layer. We use PyTorch's implementation for the RNN, GRU, and Bidirectional GRU. We also add embedding layers. PyTorch's recurrent layers naturally support multiple layers and dropout, and can be set through CLI args `--num-layers` and `--lstm-dropout`. The hidden size is set by `--hidden-size`. The defaults are 1, 0, and 256, respectively.

#	**Original Sentence**	**Modern Sentence**
1	there s beggary in the love that can be reckoned .	it would be a pretty stingy love if it could be counted and calculated .
2	poor souls , they perished .	the poor people died .
3	the propn s abused by some most villainous knave , some base notorious knave , some scurvy fellow .	the propn is being tricked by some crook , some terrible villain , some rotten bastard .
4	my best way is to creep under his gaberdine .	the best thing to do is crawl under his cloak .
5	when they do choose , they have the wisdom by their wit to lose .	when they choose , they only know how to lose .
6	her blush is guiltiness , not modesty .	she blushes from guilt , not modesty .
7	go , hang yourselves all !	go hang yourselves , all of you !
8	then , if ever thou propn acknowledge it , i will make it my quarrel .	then , if you dare to acknowledge it , i ll take up my quarrel with you .
9	and lovers ' absent hours more tedious than the dial eightscore times !	and lovers ' hours are a hundred and sixty times longer than normal ones !
10	i have no great devotion to the deed and yet he hath given me satisfying reasons .	i do n t really want to do this , but he s given me good reasons .

Figure 2: Sample Original-Modern Sentence Pairs, Processed (Without SOS/EOS Tokens)

#	**Encoder**	**Decoder**	**Attention**	**Teacher Forcing (Percent)**
1	RNN	RNN	None	None
2	GRU	GRU	None	None
3	Bidirectional GRU	GRU	None	None
4	Bidirectional GRU	GRU	Concat	None
5	Bidirectional GRU	GRU	Dot	None
6	Bidirectional GRU	GRU	Concat	0.5
7	Bidirectional GRU	GRU	Dot	0.5
8	Bidirectional GRU	GRU	Concat	1.0
9	Bidirectional GRU	GRU	Dot	1.0

Figure 3: Planned Model Experiments (Subject to change depending on results)

4.2 Decoders

4.2.1 Baseline RNN

4.2.2 GRU Decoder

4.3 Attention Mechanisms

Attention mechanisms have been shown to improve sequence to sequence translations from Bahdanau et al [1], and further work from Luong et al [4] examines global vs local approached to attention-based encoder-decoders. Common attention mechanisms are:

$$a(s_{i-1}, h_j) = \begin{cases} W_a(s_{i-1} \| h_j) & \textit{concat} \\ s_{i-1}^T \cdot W_a h_j & \textit{general} \\ s_{i-1}^T \cdot h_j & \textit{dot} \\ W_a \cdot s_{i-1} & \textit{location} \end{cases} \tag{4}$$

However, we will focus on dot attention and concat attention in our experiments.

4.3.1 Dot Attention

4.3.2 Concat Attention

4.4 Teacher Forcing

In terms of training, an encoder-decoder system can either accept the target token or the model's prediction as input during the decoding step. When we use the target token, this is known as teacher forcing, and is shown to be favored during initial trainging iterations, but should be backed off to use the model's own predicitons, as it will exhibit instability in the translations otherwise [3].

We hope to build in a system to decay the teacher forcing percentage over time, instead of our current implementation that checks a random number againist the hyperparameter.

5 Current Status and Expectation

Notation: DONE=DONE; TESTING=Works but needs verfication that it does what we want it to do; IN PROGRESS=Not fully done but ground work has been laid; TODO=No current attempt made.

Initial training is slow on a cpu, with a Bidirectional GRU Encoder + GRU Decoder + Concat Attention estimated at 11-30 minutes per 2 epochs, varying on batch size (32 and 128 tested). In order to improve training time, we have allowed an optional parameter to use a gpu. Inital run on batch size 128 yielded a training speed of 111 seconds per 2 epochs.

We will automate all planned experiments, examine the results, and use the best model as our style-transfer. We will then begin our final training session on both directions and provide results.

6 Planned Experiments

References

[1] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[2] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[3] A. M. Lamb, A. G. A. P. GOYAL, Y. Zhang, S. Zhang, A. C. Courville, and Y. Bengio. Professor forcing: A new algorithm for training recurrent networks. In *Advances In Neural Information Processing Systems*, pages 4601–4609, 2016.

[4] M.-T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[5] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[6] W. Xu, A. Ritter, B. Dolan, R. Grishman, and C. Cherry. Paraphrasing for style. In *COLING*, pages 2899–2914, 2012.

#	**Task**	**Status**	**Team Member**
1	Data Procurement	DONE	All
2	Preprocessing	DONE	Riley, Bill
3	Baseline RNN	TESTING	Riley
4	GRU Encoder	TESTING	Morris
5	Bidirectional GRU	TESTING	Morris
6	RNN Decoder	TESTING	Morris
7	GRU Decoder	TESTING	Bill
8	Attention Models	IN PROGRESS	Bill
9	Teacher Forcing	TESTING	Bill
10	Vocabulary Building	DONE	Bill
11	Train/Eval Support Code	DONE	Bill
12	GPU Support	DONE	Riley
13	Experiments	TODO	Morris, Riley

Figure 4: Current Progress

A Model Architecture Diagrams

A.1 Encoder-Decoder

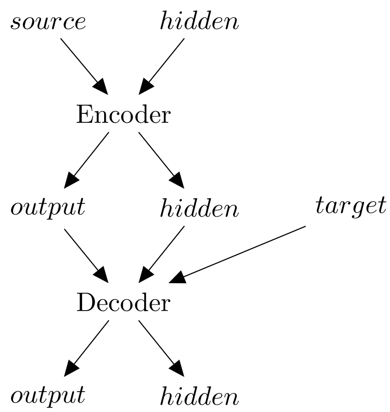

Figure 5: Model Architecture Overview for Encoder-Decoder.

A.2 Encoder

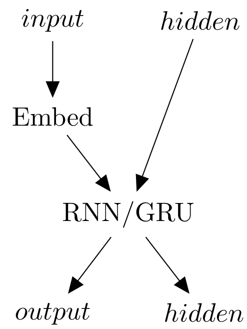

Figure 6: Model Architecture for Encoder.

A.3 Decoder

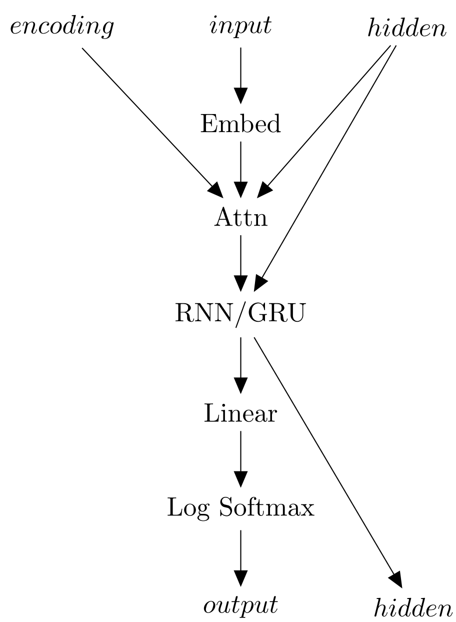

Figure 7: Model Architecture for Decoder.

A.4 Attention Mechanisms

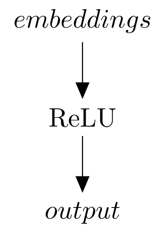

Figure 8: Model Architecture for No Attention Layer.

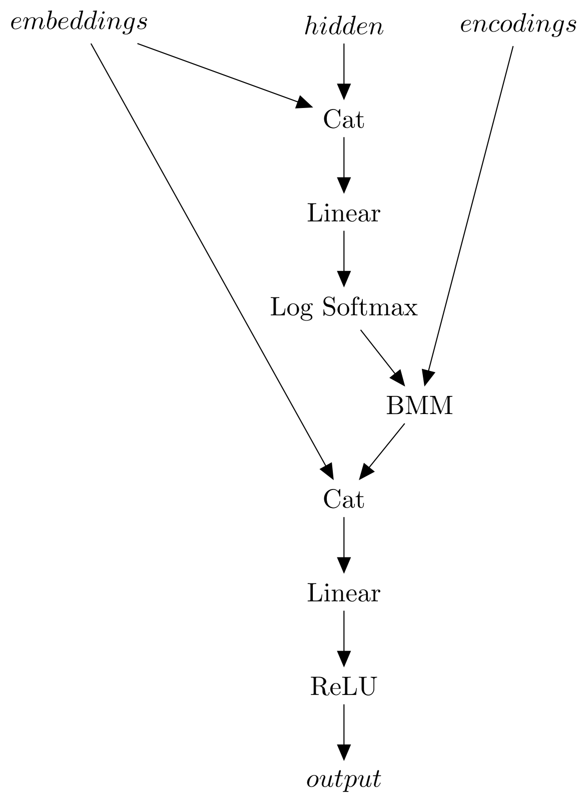

Figure 9: Model Architecture for Concat Attention Layer.