LIN205 Final Project: Implementation of Sequence to Sequence For Lyric Generation

Kevin Jesse

June 2018

1 Introduction

Sequence to sequence neural models have been a powerful method for machine translation tasks. Largely a recurrent neural network, it consists of an encoder to model the source language and a decoder to generate the translation in the target language. This method was proposed by Sustskever[16] and Cho[2]. A proposed optimization for sequence to sequence neural models was an attention mechanism that will learn to focus on an aligned word from source sentences when generating the translation word [1]. Various research to improve Bahdanau's approach has been done recently. To help with the out-of-vocabulary problem, byte-pair-encoding is created to segment words into sub-units, which can be combined to make up the unseen words in the dataset [15]. Another popular approach was to use linguistic features to improve the machine translation result [14]. For the final project of linguistics 205, I implemented Bahdanau's sequence to sequence neural model with several optimization and aligned it to a new dataset of Maroon 5 song lyrics. The results demonstrate an ability to learn and replicate portions of the artists style and lyrics. The generated lyrics show some structural dependencies that are likely from the optimization decisions.

The optimization include using attention, conditional GRU rather than GRU for decoder and improved beam search considerations

2 Dataset

The dataset was generated with bilingual machine translation tasks in mind. Typically there are a set of sentences in a language X and Y, and a model aims to learn the relationship between a source and target language. Namely, a sentence composed of tokens $\{x_i\}_{i=1}^{N} \in X$ to a sentence made of tokens $\{y_i\}_{i=1}^{N} \in Y$. While the translation is not the goal, the relationship of one lyric line to another is relevant and can be encapsulated by using the first line as the source language and the next line as the target language. Semantically, the lines should have some correlation, and structurally, the lines maintain a rhyming scheme. Thus,

the source and target machine translation dataset is constructed by offsetting the song lyric lines breaking on each song.

2.1 Data Collection and Split

To test the lyric generation model, we collected lyrics from a popular band Maroon 5. The scrapping is done with well known web scraping tool, Beautiful Soup, to scrape lyrics from all songs at metrolyrics.com. With the lyrics lines tagged and song separated, we preprocess the lyrics removing anything that is not a lyrical word. The vocabulary for both source and target language are the same so we store the same vocabulary for both encoding and decoding languages. We separate the lyrics by line and song with symbols the neural machine translator recognizes which is later tagged with the appropriate start and end characters: $< s >$, $< /s >$, and $\backslash n$. Finally, the data is sampled for 200 lyric lines for both the validation and test set. This leaves about 8500 lyric lines to be used for training. No song lyrics split across the training, test, or validation dataset.

2.2 Visualization of Embedding Space

It is important to understand our embedding space of Maroon 5 lyrics. Using the word embedding that are used for our model, we generate T-SNE plots of the entire feature space and a portion of that feature space around the word 'cold'. The shape of the embedding space in Figure 2 is likely from the fact that lyrics surround themselves with unique sets of words on a per song basis.

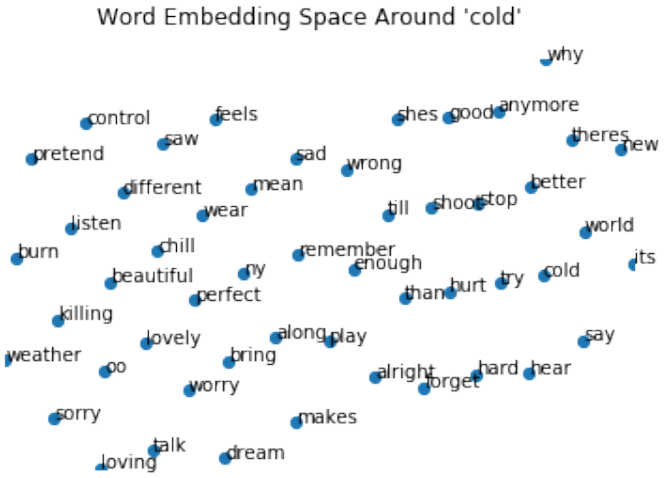

Figure 1: This is a closer look at the lyric word embedding space around the word cold

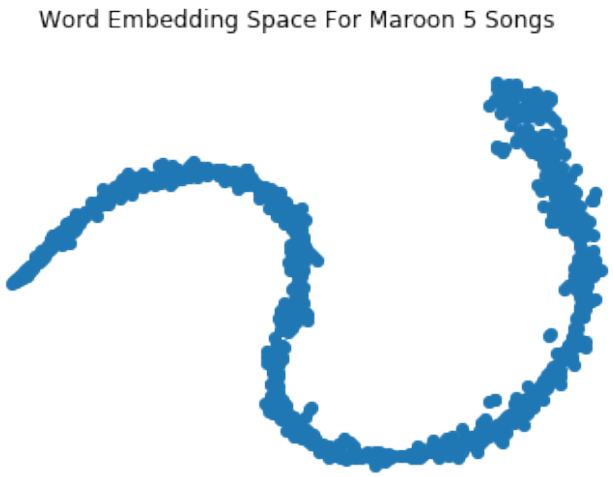

Figure 2: This is a zoomed out view of the entire embedding space.

3 Method

Prior to passing lyric sequences to the encoder, we filter sequences that have excessively long lines, greater than 80 characters. Then $< s >$, $< /s >$, and $\backslash n$ is encoded into the raw text. The lyric input is encoded into an N-length source sentence as a sequence of tokens. Each token is represented by a index value from the vocabulary space (similar to a one-hot vector). This is then mapped into the embedding space with an embedding matrix. The designed embedding space was chosen to be 128.

4 Encoder

The encoder is a bidirectional RNN [12] and processes the embedded tokens in two directions (forwards and backwards). Each each time step, the encoder generates a forward and backward hidden state vector and is concatenated together for the encoder hidden state vector. Formally, let there be hidden state vectors for each direction $\overrightarrow{h_i} = \overrightarrow{\text{RNN}(h_{i-1}, e_i)}$ and $\overleftarrow{h_i} = \overleftarrow{\text{RNN}(h_{i-1}, e_i)}$ that is concatenated as $h_i = [\overrightarrow{h_i}, \overleftarrow{h_i}]$. Through hyper parameter tuning, hidden state size was chosen to be 256.

5 Decoder

The decoder is initialized with the mean of the encoder hidden states from the representation of the source sentence or first lyric. At each time step, the

decoder generates a hidden state from the conditional GRU. To implement the conditional GRU, I followed Nemantus toolkit [13] for the conditional GRU cell implementation with attention.

5.1 Conditional GRU

The conditional GRU consists of two GRU layers with attention mechanism. At layer 1, a intermediate state is generated from the hidden state vector from the previous step and the previous embed word. Specifically, let the intermediate hidden state be $s_i = GRU_1(s_{i-1}, w_{j-1})$. The Nemantus toolkit conditional GRU calls for attention on the intermediate hidden state. The context vector that is used to capture the words the decoder should focus on when generating a token in target language Y is described as, $c_j = \sum_{i=1}^{N} \alpha_{ji} h_i$. Note that α_{ji} is computed as the softmax of a score function. The score function computes a set of values from the previous decoder hidden state and the set of encoder hidden state vectors.

At layer two we use the intermediate state generated from the first conditional GRU and the context from the attention. Formally, this value can be represented as $f_i = GRU_2(s_i, c_j)$. Now that we have the hidden state f_i, the conditional distribution of the next token y_j can be calculated with a fully connected layer L (to scale the features), and the language embedding features calculated from the conditional GRUs. By transforming the previous encoded word, the current decoded word, and the attention vector into the output embedding space, we can construct an activation a_t such that it can be mapped with the final fully connected layer and provide an output space embedding. The activation is

$$a_t = \tanh(FC_e e_{j-1} + FC_d d_j + FC_c c_j)$$

FC_* represents the mapping from the embedding space to the output space and tanh is the activation function. Combining these gives a final activation that is used in conjunction with the output of the conditional GRU fully connnected L. This output embedding and pseudo probability distribution is approximated by

$$p(y_j | y_{j-1}, x) = \text{softmax}(La_t)$$

With the softmax, the final one hot encoded vector is indexed in the vocabulary and provides us with a final word y_j.

Finally, our objective function is a simple cross entropy loss function, commonly used for label mis-classification. In our model we are using it to indicate the correct one-hot encoded selected.

Conditional GRUs with attention have been explored and demonstrated to improve traditional sequence to sequence models [5][9][7][8]. This paper/project aims to implement existing optimization and apply it in a novel manner to our artificially constructed dataset.

6 Experiment

This section explores the evaluation of the conditional GRU sequence to sequence does with the lyric dataset.

6.1 Training

The word embedding size for the encoder and decoder are 128. The encoder uses a bidirectional GRU [3] and the decoder is composed of two conditional GRUs and a fully connected layer. I used Adam [4] to optimize the model with a learning rate of .0004 and a batch size of 32; the batch size plays an important role in the stable evaluation of Adam. The gradient norm is clipped to one to avoid exploding gradients often experienced by RNNs[11]. We apply dropout at the embedding layer of the encoder, context vectors, and output layer of decoder. All weights are initialized with Kaiming's method [6]. The training learning rate decays by .2 when our evaluation metric (BLUE score) does not increase every 10 rounds. The model is tested every thousand iterations and a sample lyric is dumped for a sanity test.

6.2 Evaluation Metric

BLUE is not the ideal metric for this application because it is designed for bilingual machine translation; if this research is ever carried out further, I hope to find a better evaluation metric. However, the model returning the BLUE score provided more intelligible results than using pure loss [10]. This is expected as the BLUE score takes into consideration the quadgram probability, clips word frequency of candidate sequence, and the length normalization as to not favor shorter sequences.

7 Results

The BLUE score is surprisingly high for the model on the lyric dataset. Our model achieved a score of **24.8** BLUE score to the target language or sequential lyrics. I believe that the model achieved such a score because it learned to match portions of lyrics from various songs. While not as organic as possible, the model does seem to capture the rhyming. You can see this in 1.

The relatively good performance is likely because of the improved beam search decoding for the generation of the 'translation'. Typically beam search encoding favors sorter sentences over longer sentences, but we apply length normalization to resolve this traditional issue. The adaptive learning rate also helps the model adjust the training pace based on the performance of the validation set. The final optimization that helps is that the input to the decoder cGRU is not always the ground truth translation. We allow the model to have some chance to use the translation toke predicted from the encoder. This means our model is better simulation the same translation process as when applied to the test set. This 'noise' also helps to not overfit on the training set.

1 **Source**: *try to do what lovers do ooh*
2 **Predicted**: *love youre in my head and all of my you*

1 **Source**: *the way i used to love you oh*
2 **Predicted**: *as i wonder where were you*

1 **Source**: *and i do not understand i cannot comprehend.*
2 **Predicted**: *this love has taken its toll on me*

1 **Source**: *and im yelling and screaming cause*
2 **Prediction**: *i was not get to hear your head*

Table 1: Sample output of the sequential source and target lyrics from the neural machine translation model

7.1 Further Optimization

The sequence to sequence model could improve with a few modifications. Currently the model is restrictive on repetitive words, but often in lyrics the same word will repeat. This constraint will be removed in the next iteration of the model. Furthermore attention may not be a good solution for this specific application, as it locks in on specific parallel words. As lyrics are misaligned for translation, there is not a clear indication that attention provides an major improvement; this should be examined closer.

7.2 Project Difficulties

Originally, the plan was to adopt a handwritten sequence to sequence model on bad plots. After gathering the data, the tweets and information were poorly aligned and fixing the dataset would of required significant human computation hours. To adjust this problem, I adapted the task to Maroon 5 lyrics after seeing them live last Friday.

8 Conclusion

This paper implemented a sequence to sequence neural machine translation model from scratch with Bahdanau optimizations. We applied this high performing model on a unique dataset of Maroon 5 lyrics. The implementation performs well with coherent lyric generation and a good BLUE score. The code for this project is located on github[1]. The project served as hands on instruction for word vector embeddings, RNNs and GRUs, N-Grams, BLUE score, Encoder/Decoder architecture and more.

[1] https://github.com/kevinjesse/lyricNMT

References

[1] Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. *CoRR abs/1409.0473* (2014).

[2] Cho, K., van Merrienboer, B., Bahdanau, D., and Bengio, Y. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR abs/1409.1259* (2014).

[3] Cho, K., van Merrienboer, B., Gülçehre, Ç., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR abs/1406.1078* (2014).

[4] Cho, K., van Merrienboer, B., Gülçehre, Ç., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR abs/1406.1078* (2014).

[5] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

[6] He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR abs/1502.01852* (2015).

[7] Junczys-Dowmunt, M., and Grundkiewicz, R. An exploration of neural sequence-to-sequence architectures for automatic post-editing. *arXiv preprint arXiv:1706.04138* (2017).

[8] Kiros, R., Zhu, Y., Salakhutdinov, R. R., Zemel, R., Urtasun, R., Torralba, A., and Fidler, S. Skip-thought vectors. In *Advances in neural information processing systems* (2015), pp. 3294–3302.

[9] Libovický, J., and Helcl, J. Attention strategies for multi-source sequence-to-sequence learning. *arXiv preprint arXiv:1704.06567* (2017).

[10] Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics* (Stroudsburg, PA, USA, 2002), ACL '02, Association for Computational Linguistics, pp. 311–318.

[11] Pascanu, R., Mikolov, T., and Bengio, Y. Understanding the exploding gradient problem. *CoRR abs/1211.5063* (2012).

[12] Schuster, M., and Paliwal, K. Bidirectional recurrent neural networks. *Trans. Sig. Proc. 45*, 11 (Nov. 1997), 2673–2681.

[13] Sennrich, R., Firat, O., Cho, K., Birch, A., Haddow, B., Hitschler, J., Junczys-Dowmunt, M., Läubli, S., Barone, A. V. M., Mokry, J., et al. Nematus: a toolkit for neural machine translation.

[14] Sennrich, R., and Haddow, B. Linguistic input features improve neural machine translation. *CoRR abs/1606.02892* (2016).

[15] Sennrich, R., Haddow, B., and Birch, A. Neural machine translation of rare words with subword units. *CoRR abs/1508.07909* (2015).

[16] Sutskever, I., Vinyals, O., and Le, Q. V. Sequence to sequence learning with neural networks. *CoRR abs/1409.3215* (2014).