Natural Little Ponies: Final Report
CSE 481 N

Nelson Tan, Brandon Fok, Sam Wheelock

June 2019

Abstract

For our natural language processing capstone project we created a music lyric generator with the main differentiator from existing neural network based lyric generators being the ability to generate rhyming lyrics. With this, we aim to help lyricists overcome writer's block by providing inspiration in the form of computer-generated lyrics. By inputting their subject, song structure (Chorus, Verse, Bridge, etc.), and rhyming scheme for each section (ie: ABAB), the user receives a complete song with their given criteria. An evaluation with human responses showed that decent lyrics, although not always up to a lyricist's standards, can be generated given these inputs.

1 Introduction

Writer's block is an adversity every musician has faced. This paper presents an attempt at using text generation and text completion as a means of combating this problem. We use a combination of an LSTM model (long short-term memory) along with post-processing methods to incorporate rhyming in our output.

Lyric generation is a difficult task to automate. Most studies use recurrent neural networks (RNN) Sherstinsky [2020], as they retain context (and therefore writer's intent, or topics) well. Zhe Wang [2016] introduced a novel poetry generating method which used a two-stage technique using a RNN enconder-decoder framework - first taking in the writer's intent, then generating lines based on this. Zhang and Lapata [2014] also used an RNN-based approach in which they use a combination of an RNN and language model to learn the representation of previously generated lines, and use that as input to generate the next line. Because of the success previous papers have had with using RNNs for lyric generation, we also use an RNN-based approach to this task.

Evaluation is difficult to automate for lyrics. While some papers have tried to utilize automated evaluation methods (Zhang and Lapata [2014] used BLEU-based evaluation), we found that given our lack of time and the amount of engineering needed to develop automated evaluation, we needed to find a way to manually evaluate our results. We found the evaluation technique from Zhe Wang [2016] and Zhang and Lapata [2014] to be both simple and effective. Both papers used an evaluation survey which they sent to human participants who had to judge the given generated poem based on four standards: Poeticness, Fluency, Coherence, and Meaning. We used a similar evaluation technique on Fluency, Coherency, Topic, and Rhyming.

Our results show that simple LSTM models, with help from post-processing methods that enforce rhyming, has potential to generate creative lyrics and inspire writers.

2 Terminology

Intro, Verse, Chorus, Bridge, Outro: Five sections that a song is typically split into. Each section has distinct style and rhyming patterns.
Minecraft: The worlds 2nd most played video game (1st: Tetris), the subject of many songs and parodies.
Perfect Rhyme: words with the exact same vowel and consonant sounds at the end (e.g. cat, fat)
Imperfect Rhyme: words with same vowel or consonant sound in the same part of the word (e.g. cat, pad)

3 Technical Ideas

3.1 Datasets and Pre-processing

We created our own datasets using the Genius API[1]. Our data consisted of four different datasets, each pertaining to a musical genre: Pop, Hip Hop, Country, and Minecraft. We were able to get songs in each genre that were neatly sectioned by Intro, Verse, Chorus, Bridge, and Outro giving us the ability to generate a full song. For Pop, we took the top 40 songs from each of the top 80 artists of 2019. For Hip Hop and Country, we did the same with the top 30 artists of 2019. For Minecraft, we were only able to find 6 artists. The purpose of the Minecraft dataset was to explore the quality of a text generation model trained on a very specific and focused set of lyrics. This would give us insight on the viability of training models to a specific topic.

3.2 Training

We used a Python library called TextgenRNN[2] to train our models. We used a 2-layer, 128-cell word-level LSTM for each section of each dataset (twenty models total). These parameters, as well as the default hyperparameters supplied by TextgenRNN for word embeddings and vocabulary size gave us a good middle-ground for time to train versus quality of the model.

3.3 Post-Processing

To implement the rhyming scheme into our generation, we utilized a rhyming API called Phyme[3]. This allowed us to group each line into their own rhyming category. We continued to generate new lines and group them until we had enough lines for the specified rhyming scheme.

A noticeable problem with this method of post-processing rhymes is that the lines would be swapped around, thus lowering the coherency of the generation. In order to combat this, we sorted our lines based on their distance from other lines in their rhyme group to take advantage of long short-term memory.

When a user specified a topic, we simply made the topic the prefix of our generations, hoping that the model would maintain the topic throughout generations. While this somewhat worked, there is room for improvement here.

4 Evaluation Technique

The goal of our project is to answer the following questions: Can we generate lyrics of an entire song that people believe are real and can we add rhymes into these lyrics while maintaining fluency and coherency?

To answer these questions we created an evaluation that scored our generated lyrics and real lyrics on the following metrics: Fluency (is each individual line grammatically correct and smooth when read?), Coherency (does the entire section read smoothly and follow the same theme/topic?), Following the Topic (do the lyrics follow the given topic?), and Rhyming (do the line endings rhyme?). We used a survey to rate each of these metrics between 1 - 5, where 1 meant that the lyrics did not satisfy the metric and 5 meant that

[1] https://docs.genius.com/

[2] https://github.com/minimaxir/textgenrnn

[3] https://github.com/jameswenzel/Phyme

	Fluency (1-5)	Coherency (1-5)	Follows Topic (1-5)	Rhymes (1-5)
Real Songs	4.3	4.4	4.5	3.9
Fully Generated Songs	2.1	2.1	3	2.8
Generated Chorus	3.3	3.3	N/A	2.9

Table 1: Average ratings for our songs after surveying 28 people

People who write lyrics	Fluency (1-5)	Coherency (1-5)	Follows Topic (1-5)	Rhymes (1-5)
Real Songs	4.4	4.4	4.6	3.2
Fully Generated Songs	2	1.7	2.7	3
Generated Chorus	3.2	3.2	N/A	3.2

Table 2: Average ratings for our songs from 5 song writers

they followed the metric extremely well. The survey contained three generated songs and one real song so we could average the scores across multiple generations. We also added a question to our survey asking for the survey participant to guess which songs were generated versus which songs were not.

4.1 Evaluation Results

We had participants rate a real song, three generated full songs with the same topic and rhyming scheme as the real song, and one hand-picked generated chorus. The results can be seen in Table 1.

In our survey we gathered information about each participants musical expertise. We received 5 responses from people who write lyrics so we decided to see if their averages were any different. See Table 2 to see the results.

We also included in our survey a question to see if we were able to fool our participants. See Table 3 to see the percentage of people who believed the songs were generated.

4.2 Analysis of Results

From our results, it is clear that our generated lyrics do not perform as well as real lyrics in any of the four metrics. We can also see that our fully generated songs performed worse than our generated chorus which was expected since we hand-picked one of our better choruses.

With our chorus, 36% of the surveyors believe the chorus was real, while only 13% believe that the fully generated songs were real. We believe this difference is due to the grammatical errors that were often introduced into our generations. Longer generations were more likely to have obvious grammatical errors.

Averaging the results from lyric writers gave insightful results from those who judge lyrics regularly. They rated rhyming more evenly between both the real lyrics and the generated lyrics, perhaps due to the

% of people that believed the song was generated	
Real Songs	7%
Fully Generated Songs	87%
Generated Chorus	64%

Table 3: Percentage of survey participants that believed the songs were generated

fact that our rhyming algorithm uses imperfect rhymes. Lyric writers may be able to see that these imperfect rhymes are in fact rhymes. They were also better at identifying generated songs and gave lower rating to generated songs.

5 Related Work

Dhariwal et al. [2020]: Used raw MIDI data from songs to generate fully completed songs with lyrics and audio.

6 Future Work and Limitations

1. **Grammar** The grammar from our generations was not always correct. We assume these errors occur because our models were not trained on a large enough data set and that we did not do enough hyperparameter tuning. In the future, we want to train on top of a pre-trained model on song lyrics or use a lot more data.
2. **Rhyming** Another area to look into would be building rhyming into the model instead of post processing the rhymes. This would require much more research but would both speed up the generations and better maintain coherency across lines. Malmi et al. [2016] made a Rap Lyric Generator that only trained on the last word of each line. When generating lyrics, it "plagiarizes" each line but the model predicts the next line based on how likely it is to rhyme with the previous line.
3. **Plagiarism** Plagiarism became a big problem for us, especially with our smaller datasets (ie: Minecraft dataset). A plagiarism detection algorithm would solve solve this problem. We were able to significantly improve our model through post-processing, data gathering, and hyperparameter tuning but time constraints did not allow us to implement the three features stated above, particularly rhyming without post-processing.

7 Conclusion

Our models performed decently after sifting through a few runs of the lyric generator and experimenting with the hyperparameters. While the generations did not fool everyone, we managed to figure out an effective way of incorporating rhymes into our generations. Although the rhyming algorithm is not perfect, it performed better than online neural network based lyric generators albeit much slower. Video Presentation: https://drive.google.com/file/d/1PHzB4Sb0SyEnPlAXNGr70Wzm5NXMyEbQ/view?usp=sharing

References

Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music, 2020.

Eric Malmi, Pyry Takala, Hannu Toivonen, Tapani Raiko, and Aristides Gionis. Dopelearning. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug 2016. doi: 10.1145/2939672.2939679. URL http://dx.doi.org/10.1145/2939672.2939679.

Alex Sherstinsky. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 404:132306, Mar 2020. ISSN 0167-2789. doi: 10.1016/j.physd.2019.132306. URL http://dx.doi.org/10.1016/j.physd.2019.132306.

Xingxing Zhang and Mirella Lapata. Chinese poetry generation with recurrent neural networks, October 2014. URL https://www.aclweb.org/anthology/D14-1074.pdf.

Hua Wu Haiyang Wu Wei Li Haifeng Wang Enhong Chen Zhe Wang, Wei He. Chinese poetry generation with planning based neural network, December 2016. URL https://arxiv.org/pdf/1610.09889.pdf.