Rap Lyric Generation using Markov chain and LSTM

Nidhi Sharma, Satwik Kashyap and Prof. Nick Brown

Information Systems program, Northeastern University, Boston, MA 02115

Abstract

This paper demonstrates the efficiency of a Long Short-Term Memory Language model along with the baseline model to generate unconstrained rap lyrics. This model aims at creating similar in style yet unique lyrics. Additionally, the model defines the line length of the lyrics by comparing the last word with the most rhymed word. The experiments show that automatically generated lyrics can correctly capture the patterns and styles of multiple lyricist and fit into certain scenarios. As evaluation method can be used to analyze system performance, we propose methods like Cosine similarity, Rhyme Density and bilingual evaluation understudy. We provide a corpus of lyrics for 11 rap artists with 11,540 unique words and 20,007 lines of lyrics. We also suggest further work for understanding and improving this process and generating more novel lyrics.

1 Introduction

Rap lyrics are often stigmatized as offensive and inappropriate, however, it coins up most of the words of the vocabulary. William Shakespeare is credited by the Oxford English Dictionary for coining up to 2,000 words, which are now out worded by most modern-day rappers containing 7,392 unique words compared to the celebrated poet who only had 5,170. [1]

These statistics suggests that rap lyrics have larger vocabulary compared to the English literature. Thus, it is significantly important to develop an artificial intelligence network that is capable of suggesting vocabulary of such magnitude in order to address the writer's problem. The goal is to present something in a style that is believable enough to be credited to the performer, that is, create a network that can take any artist's lyrics and generate similar yet unique lyrics.

To accomplish this, a language model is created to produce the text, while also understanding what 'style' means in a quantitative sense. This paper examines the generated rap lyrics on Long Short-Term Memory (LSTM) network with Markov as the baseline model and assess its quality through similarity score, rhyme density score, bilingual evaluation understudy (BLEU) Score, and human evaluation. Additionally, this paper analyzes two scenarios: (1) addressing the writer's block problem by producing similar yet unique lyrics (2) assess the performance of the model using computational evaluation methods.

2 Related Work

Natural Language Generation is a rapidly evolving field of natural language processing. Recent work conducted on Recurrent Neural Network (RNN) has shown effective results for text generation [2]. In this study, RNN was used to create a language model at character-level. Their results show that the network learns grammatical and punctuation rules, such as opening and closing parentheses along with the English vocabulary at character-level. Another study conducted by Graves (2013) [3] used a variation of RNN, that is, LSTM network for text generation, handwriting synthesis and the results were quite impressive while able to generate highly realistic cursive handwriting. This study showed that LSTM performed better than RNN on sequential

data. A similar study by Tikhonov(2018) [4] used a LSTM with extended phonetic and semantic embeddings for stylized poetry generation. The quality of the resulting poems generated by the network is estimated through BLEU score. Their experiments show that the proposed model consistently outperforms random sample and vanilla-LSTM baselines, humans also tend to associate machine generated texts with the target author. "Long Short-Term Memory" neural networks have also been executed by Potash et al. [5], whose objective was to generate rap lyrics in the similar style of a given rapper but not identical to his existing lyrics, emulating the task of ghostwriting. They compared their work with the model of Barbieri et al. [6], which employed constrained Markov processes on the same task). Their work shows how this approach can be used for the semi-automatic generation of lyrics in the style of a popular author that has the same structure as an existing song.

In regard to rap lyrics, Wu et al. (2013) [7] present a system for rap lyric generation that produces a single line of lyrics that are meant to be a response to a single line of input. The work in this paper is similar to that of Malmi et al. (2015) [8], where the authors create fixed 16-line verses, generating the verse line-byline using full lines from existing rap songs. The system predicts the best next line based on the previous lines, using a system that records an 81.9% accuracy predicting next lines in already existing verses. The feature that provides the greatest accuracy gain is a neural embedding of the lines, created from the character level.

Similar work has been done in Xing Wu(2018) [9] study where LSTM network is used to process and generate the next line word by word. A hierarchical attention model is designed to capture the contextual information at both sentence and document level, which could learn high level representations of each lyric line and the entire document. Their results show that the automatically generated lyrics can correctly capture the patterns and styles of a certain lyricist and fit into certain scenarios.

To quantify the critical aspect of their experiment, which is to produce similar yet different lyrics, they evaluated their models by correlating the cosine similarity between the existing and generated lyrics using the "Inverse Document Frequency" algorithm, as well as, computing its "Rhyme Density" score.

However, this paper aims at analyzing and comparing the results using a more recent RNN called "Gated Recurrent Unit" and evaluate the models using cosine similarity between the input and the generated lyrics, rhyme density, Bleu score, spaCy, cross-entropy and human evaluation.

3 Generating Lyrics

3.1 LSTM (Long Short-Term Memory)

LSTM is a type of Recurrent Neural Network with a different architecture. The foundation of an RNN is a word embedding E that provides a vector representation for each of the words in our corpus. Given k number of words 'w', the probability of RNN is defined as:

$$P(w_{k+1}|w_k, \ldots., w_0; E, \phi) = f(x, s)$$

At each time-step the RNN computes 'f' given an observation 'x' and a previous state 's'. The input goes through a transformation where it passes through one or several hidden layers.

LSTM model contains a 'memory cell' that can maintain information in memory for long periods of time. LSTM contains forget gate, input gate, output gate and cell state where

each of them has its own bias vector, and the hidden layer at each time-step is a complex non-linear combination of gate, cell and hidden vectors.

3.2 Verse Structure and Rhyme Inference

The goal of the model is to not just generate lyrics but generate the structure for the lyrics as well. This has been achieved using a LSTM network and training it with the structure of the rap.

3.2.1 Rhyming

There are different types of rhymes, such as, *Perfect Rhyme*, *Assonance*, *Alliteration*, *Consonance*, *Multisyllable* but the Perfect rhyme is the most basic kind and is the dominant form of rhyme in the early days of rap. Now it is one of a variety of rhyme types but is still widely used because of the strong connection it creates in lyrics. In a perfect rhyme, the words share exactly the same end sound, as in "slang – gang,".

3.2.2 Song structure

A typical rap song follows a pattern of alternating verses and choruses. These in turn consist of lines, which break down to individual words and finally to syllables. Usually, one bar contains 4 counts and each count consists of 1-line. Verses, which constitute the main body of a song, are often composed of 16 lines.

Consecutive lines can be joined through rhyme, which is typically placed at the end of the lines but can appear anywhere within the lines. The same end rhyme can be maintained for a couple of lines or even throughout an entire verse.

3.2.3 Automatic rhyme detection

The aim of this experiment is to automatically detect perfect rhymes from lyrics given as text. This is done by first obtaining the rhyming words for the last word of the line by using *Pronouncing* library of Python. Using *pronouncing.rhymes(word)*, the authors notice that most of the rhymed words end with the same letters due to which we only consider the last two characters of the rhymed words. Then, to obtain the similar sounding words in proper order, only the words occurring the most in the list are taken into account, reversed and sorted to get the words in alphabetical order and reversed once again to finally obtain words with similar sound in order.

These rhymed words are arranged in an order and the goal is to determine how close the rhyming words are to each other. The closer the words, the similar will be their score with respect to each other. To compute the score, the index of each word is taken and divided by the length of all the rhymed words in the list. Hence, words like *rap* and *gap* will have a similar score.

3.2.4 Syllable structure

A syllable is a single, unbroken sound of a spoken (or written) word that contain vowel and consonants. The most important part of the structure of our rhymes is the syllable structure because each line will create a beat in our rap. Drawing out too many or too little syllables in a single line can disrupt the beat of our rap due to which we need to balance the rhyming lines by having the same count of syllables for each line.

For this experiment, the value for the maximum allowed syllables in a line has been calculated by taking the average of the number of syllables in each bar. The network will make sure to restrict our syllables to the average syllable count to maintain the beat of the rap.

4 Implementation

4.1 Data

Data was crawled from two websites to compile a list of 10 popular English-speaking rap artists and scraped all their songs. In total, 17,505 lines and 145,597 words from 250 different songs were collected. The lyrics were preprocessed by removing the empty strings, extra unnecessary punctuation and whitespace, and lowercasing all the words.

4.2 Baseline Model

Makov chain is used in this experiment for lyric generation that works by producing sentences based on recombination of elements of history of known sentences to generate meaningful sentences.

The goal is to make an unsupervised system, that is, not using any constraints to generate the lyrics. Thus, the baseline model simplifies to a basic n-gram model. Given, $w_{k+n-1}, \dots, w_k$ words, the system generates a new token t:

$$P(w_{k+n} = t | w_{k+n-1}, \dots, w_k) = \frac{|w_k, \dots, w_{k+n-1}, t|}{|w_k, \dots, w_{k+n-1}, \bullet|}$$

Where $|w_k \dots w_{k+n-1} t|$ is the amount of times the context $(w_{k+n-1,} \dots, w_{1)}$ is followed by t in the training data and $|w_k \dots w_{k+n-1} \bullet|$ is the amount of times the context appears followed by any token.

There is the possibility that the context has never been encountered in the training data. When this occurs, a smaller n-gram model is selected:

$$P(w_{k+n} = t | w_{k+n-2}, \dots, w_k) = \frac{|w_k, \dots, w_{k+n-2}, \bullet, t|}{|w_k, \dots, w_{k+n-2}, \bullet, \bullet|}$$

4.3 Model Initialization

4.3.1 LSTM implementation

A Python implementation of an LSTM built on top of Keras is used in this experiment. The amount of LSTM inputs/outputs are set equal to the vocabulary size, input shape as (2,2) with 14 hidden layers. For this experiment, the Learning rate is 0. 001, loss function is Mean Squared Error and optimizer is Stochastic gradient descent.

The hidden state of an LSTM cell is calculated as:

$$i_t = \sigma(x_t U^i + h_{t-1} W^i)$$

$$f_t = \sigma(x_t U^f + h_{t-1} W^f)$$

$$o_t = \sigma(x_t U^o + h_{t-1} W^o)$$

$$\check{C}_t = \tanh(x_t U^g + h_{t-1} W^g)$$

$$C_t = \sigma(f_t * C_{t-1} + i_t * \check{C}_t)$$

$$h_t = \tanh(C_t) * o_t$$

4.3.2 Next Line Prediction

Consider a large repository of rap lyrics, which is treated as training data, and is used to learn a model between consecutive lines in rap lyrics. Then, selecting a random line to start from the corpus, the trained model is used to predict the next consecutive lines. The output predicted is recursively used to predict the next line to give a model prediction for the next 100 lines.

The method can then be used to construct a song line-by-line, by selecting the most relevant lines from Markov generated sentences based on the model predicted output.

5 Evaluation Methods

It is difficult to judge the quality of the generated lyrics using human evaluation, especially on large scale studies that use a full dataset of 30 different rap artists. Therefore, it is necessary to automate the evaluation method which can capture the main aspect of it, that is, similar yet unique lyrics.

5.1 Similarity using cosine distance

An algorithm proposed by (Mahedero et al., 2005)[10] is used for calculating the similarity between produced lyrics and all verses from the same artist. This algorithm is based on the well-known Inverse Document Frequency, using cosine on document vectors to calculate distance. First, we build the Term-Document Matrix with weights for each term in each song:

$$w_{ij} = f_{ij}\log(\frac{N}{n_j})$$

where N is the total number of documents, nj is the number of verses that contains term j and fij is the frequency of term j in the ith verse.

Using this matrix, the cosine distance between verses is calculated and used as a measure of similarity. Here similarity is the max similarity: of all verses it is most similar to. The lower the max similarity score, the more novel the lyrics.

5.2 Stylistic Similarity via Rhyme Density

Rhyme density method proposed by Hirjee and Brown (2010a)[11] is used to evaluate how well the generated verse models an artist's style. The goal is to produce rhyme types and rhyme frequency similar to the target artist.

In order to fully automate this method, this study proposes to handle highly repetitive text by weighting the rhyme density of a given verse by its entropy. More specifically, for a given verse, entropy is calculated at the token level and divided by the total number of tokens in that verse. Verses with highly repetitive text will have a low entropy, which results in down weighting the rhyme density of verses that produce false positive rhymes due to their repetitive text.

5.3 BLEU Score

BLEU is a score for evaluating a generated sentence to a reference sentence, that is, to evaluate text generated of natural language processing tasks. A perfect match results in a score of 1.0, whereas a perfect mismatch results in a score of 0.0.

5.3.1 Sentence BLEU Score

Natural Language Toolkit (NLTK) provides the sentence_bleu() function for evaluating a candidate sentence against one or more reference sentences. The reference sentences must be provided as a list of sentences where each reference is a list of tokens.

5.3.2 Corpus BLEU Score

NLTK also provides a function called corpus_bleu() for calculating the BLEU score for multiple sentences such as a paragraph or a document. The references must be specified as a list of documents where each document is a list of references and each alternative reference is a list of tokens.

6 Experiment results

In this work, the effectiveness of an LSTM model is shown for generating novel lyrics that are similar in style yet different from a target artist. The results of this experiment show that, as an unsupervised, non-template model, the LSTM model is able to produce novel lyrics with the perfect rhyme.

Below is a sample of lyrics generated:

'i swear im the pile in the sheets',
'theres a joker on the floor seats',
'smoked out like sanford snippits',
'to hide the goods i would throw fits',
'streets are seedy i call the shots',
'but at least purposely',
'to the top so lonely',
'flow got spring like a whirlwind phase',
'you get him and fuck you tahm bout',

'you feeling like a strikeout',
'im up way too much bet uncut',

In terms of evaluation, the cosine similarity score that is used to measure how similar the documents are is higher than expected, that is, 0.74. Since rap has a big vocabulary, the chances of the same words being repeated in the documents are high. As the size of the corpus increases, the number of words tend to increase in output even if the generated lyrics talks about different topics. Due to high vocabulary, the similarity score increases with words appearing in different context.

The rhyme density metric is difficult to calculate as there are certain artists that distinguish themselves by having more complicated rhyme schemes, such as the use of internal or polysyllabic rhymes. There are different types of rhymes in a rap, that is, perfect rhyme, assonance, multisyllable rhyme, alliteration which are present either at the end of a line or in between lines. In this experiment, perfect rhyme score is calculated having value of 13%.

Bilingual Evaluation Understudy (BLEU), is a score used to evaluate text generated for a suite of natural language processing tasks. The Bleu score calculated in this experiment is 1.7710*e(-232)

Jaccard similarity score, used to represent the similarity between two documents has also been calculated to 0.0230. A value “0” means the documents are completely dissimilar, “1” that they are identical, and values between 0 and 1 representing a degree of similarity.

Figure 1 shows Mean Squared error as a loss function curve with respect to the number of epochs.

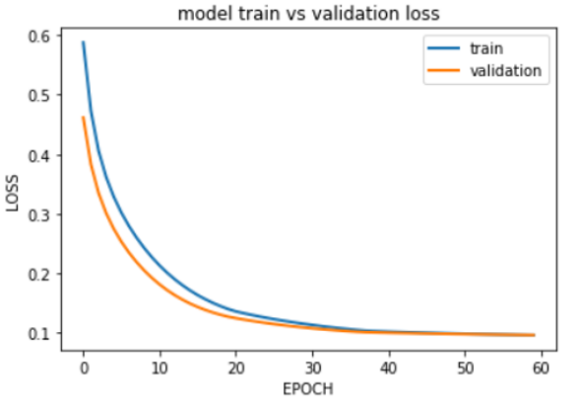

Figure 1: validation loss and number of epochs for training

Figure 2 highlights the Root Mean Squared value for the training and validation dataset.

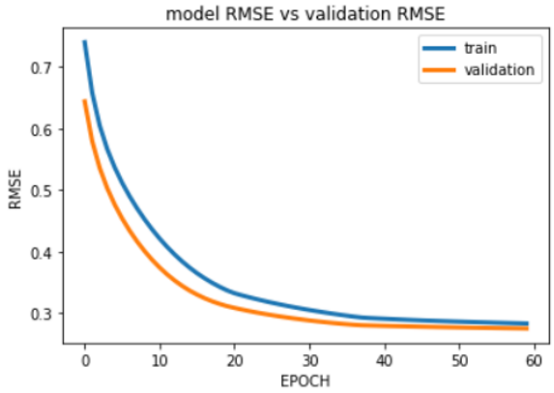

Figure 2: RMSE score and number of epochs in validation

7 Discussion

Text generation is widely regarded as the highly difficult problem, partially because of the lack of large dataset. Baotian Hu(2015) [12] showed in their work the promising result using an RNN. The corpus consists of over 2 million real Chinese short texts with short summaries given by the author of each text. Similarly, in order to come up with more novel rap lyrics, the LSTM can be fed with a large corpus size. Unfortunately, the corpus size in this experiment is not large enough.

Better results can be expected on resolving this problem.

It is difficult to reflect a particular genre with the dataset used in this experiment. This can be approached with Unsupervised learning method, that is, clustering where song lyrics of similar rhyme scheme will be grouped together and can be further used to generate rap reflecting a particular style.

Transfer Learning was used in the work of Katy Ilonka [13] demonstrating how pre-trained language models can be effectively tuned with small data sets to generate Style-Specific text. As current methods require large data sets to generate coherent sentences, which severely limits their creative potential given that the majority of stylistic literary data sets are relatively small. Their work empirically shows the effectiveness of this method across three distinct literary styles where only a small (e.g. less than 6k) number of word tokens are available.

8 Conclusion

In this paper, the effectiveness of an LSTM model for generating novel lyrics is analyzed so that the lyrics are similar in style to a target artist. We use the LSTM model with a baseline Markov model and evaluate our model using Rhyme Density, Cosine similarity and BLEU score. The results of our experiments show that, as an unsupervised, non-template model, the LSTM model is better able to produce novel lyrics.

References

[1] M, D. (2018). *The Largest Vocabulary in Hip Hop.*[online] Available at: https://pudding.cool/2017/02/vocabulary/ [Accessed 5 Aug. 2018].

[2] Sutskever, I., Martens, J. and Hinton, G. (2013). *Generating Text with Recurrent Neural Networks*. University of Toronto.

[3] Graves, A. (2013). *Generating Sequences With Recurrent Neural Networks*. University of Toronto.

[4] Tikhonov, A. and Yamshchikov, I. (2019). *Multilingual Approach for the Automated Generation of Author-Stylized Poetry*. Max Planck Institute for Mathematics in the Sciences.

[5] Potash, P., Romanov, A. and Rumshisky, A. (2015). *GhostWriter: Using an LSTM for Automatic Rap Lyric Generation*. Dept. of Computer Science. University of Massachusetts Lowell.

[6] Barbieri, G., Pachet, F., Roy, P. and Esposti, M. (2012). *Markov Constraints for Generating Lyrics with Style*.

[7] Wu, D., Addanki, K., Saers, M. and Beloucif, M. (2013). *Learning to Freestyle: Hip Hop Challenge-Response Induction via Transduction Rule Segmentation*. Department of Computer Science. HKUST, Clear Water Bay, Hong Kong.

[8] Malmi, E., Takala, P., Toivonen, H., Raiko, T. and Gionis, A. (2015). *DopeLearning: A Computational Approach to Rap Lyrics Generation*. Aalto University.

[9] Wu, X., Du, Z., Guo, Y. and Fujita, H. (2018). *Hierarchical attention based long short-term memory for Chinese lyric generation*. Shanghai Institute for Advanced Communication and Data Science.

[10] Mahedero, J., Mart´ınez, A. and Cano, P. (2005). natural language processing of lyrics. Spain.

[11] HIRJEE, H. and BROWN, D. (2010). *Using Automated Rhyme Detection to Characterize Rhyming Style in Rap Music*. Cheriton School of Computer Science. University of Waterloo.

[12] Hu, B., Chen, Q. and Zhu, F. (2018). *LCSTS: A Large Scale Chinese Short Text Summarization Dataset.* Graduate. Harbin Institute of Technology, Shenzhen Graduate School.

[13] Gero, K., Karamanolakis, G. and Chilton, L. (2018*). Transfer Learning for Style-Specific Text Generation.* Computer Science Department. Columbia University.