Ghostwriting Rap Lyrics using Character Level LSTM

Deepak Sharma
Dept of Computer Science
IIT Delhi, India
mcs172071@iitd.ac.in

Harish Chandra Thuwal
Dept of Computer Science
IIT Delhi, India
hthuwal@gmail.com

Abstract

This work presents a Character Level LSTM language model for rap lyrics generation. The aim of this model is to generate unconstrained rap lyrics specific to an artist's style, but different than existing lyrics.

1 Introduction

Ghostwriting is the task where content for an artist is created by someone else, but the official credit for the content is given to the artist. The content should be similar to the style of the artist. Ghostwriting is done in different domains - music,literature,and even nowadays for internet blogs. In the domain of rap lyrics, a ghostwriter writes lyrics for an artist, matching his style - rhyme density, vocabulary etc.

For lyrics/language generation we need a language model. A language model provides a probability distribution over the sequence of words/characters. Usually it is used in context with other tasks where it predicts the most probable alternative sentence. For language generation, words(characters) are generated conditionally on the previous partial generated output.

While an n-gram model would only be able to condition on a finite history, Recurrent Neural Networks (RNN) on the other hand theoretically can remember all the prediction history. Long Short Term Memory (LSTM) is a variant of RNN that is able to better handle long range dependencies. A character level LSTM generates the language text on a character by character basis. It is known to be better at word shapes and OOV (out of vocabulary) words.

(Potash et al., 2015) have shown a word level LSTM language model to be effective than an ngram model for the task of rap lyrics generation. It consists of a 2 layer LSTM with an embedding layer, decode layer and a softmax layer (at each time step) to generate probability distributions over words in vocabulary. The model is trained so as to minimize the KL divergence loss of each output word from the corresponding training word's one hot probability distribution.

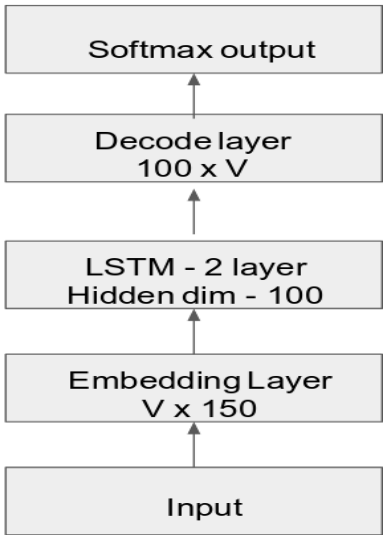

Figure 1: Model by (Potash et al., 2015)

This work shows that similar if not better results can be obtained using character level LSTM than a vanilla LSTM over words for the task of ghostwriting.

2 Related work

Earlier approaches to language generation were based on n gram models (Barbieri et al., 2012). In recent years, Recurrent Neural Network(RNN) models have been effectively used for language modeling. (Sutskever et al., 2011) used a character

level RNN which effectively captured many grammatical and punctuation rules. (Graves, 2013) showed LSTM to model language better than a regular RNN.

For rap lyrics generation, (Wu et al., 2013) provide a model which works as a request response model , where a single line is generated in response to a single line of input. (Malmi et al., 2016) provide a information retrieval based model where lyric verses are produced by directly using lines from the training corpus. The model uses a ranking based algorithm that generates a set of candidates for next line. Human engineered features as well as semantic features generated by a Deep Neural Network are used. Demo can be found at http://deepbeat.org/.

3 Model

We propose a character level LSTM model to generate lyrics. Given current letter and all previous letters, the model tries to predict the next character of the sequence. The model produces an output vector o = $(o_1, o_2, ...o_n)$ for every training sequence of characters $(c_1, c_2, ...c_n)$. $softmax(o_t)$ will give the predictive probability distribution for $P(x_{t+1}|x_{1..t})$. Thus the model aims to maximize the total log probability of the training data. During training for each sequence we use all its characters except the last one as an input and the same sequence starting from the second character as ground truth.

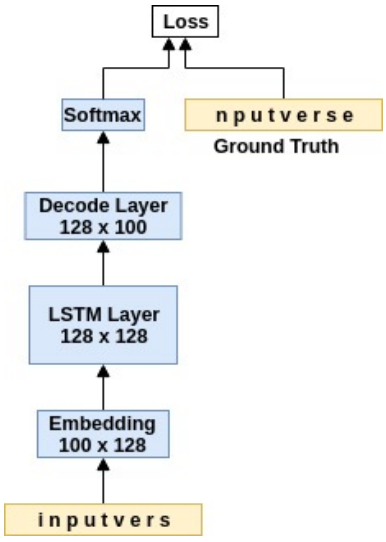

Figure 2: char level LSTM Model

Our model consists of embedding layer, a single lstm layer and a decode layer. Each character is represented by a 128 dimensional embedding and we have about 100 different printable characters. So the size of the embedding layer is 100 x 128. The size of the lstm layer and the decode layer are 128 x 128 and 128 x 100 respectively.

Since the output of the model is the conditional distribution $P(x_{t+1}|x_{1..t})$. We can generate lyrics by sampling the next character from this distribution and then feeding it back as the next input. Repeat this process until we either sample a character denoting *endofverse* or reach the maximum length.

4 Experiments

4.1 Dataset

To compare with the baseline model (Potash et al., 2015), we used the same data set that they used. The data set consists of 191 rap songs by the artist Fabolous, from which 221 verses were selected whose length were greater than 175 words. All of the data set is used for training the char level LSTM model.

We also used the `https://www.kaggle.com/mousehead/songlyrics` dataset, which consists of about 55,000 songs from different artists to pretrain the model and learn better embeddings as the Fabolous dataset is too small in size.

4.2 Evaluation metrics

For evaluating the model for the task of ghostwriting, we need two evaluation metrics : rhyme density (for the style of artist) and maximum similarity (similarity to existing lyrics).

Rhyme density is obtained as the ratio of number of rhymed syllables and total number of syllables. A good model should produce lyrics that have rhyme density similar to that of the artist. We have used a tool by (Hirjee and Brown, 2010) to calculate the rhyme density of generated lyrics. For the artist Fabolous, it is found that the average rhyme density is 0.34.

To determine the novelty of generated lyrics, similarity of the generated lyrics is calculated with respect to the existing lyrics in the dataset. For similarity calculation , cosine similarity is used on the TF-IDF document(verse) vectors. Cosine similarity of the generated verse with every existing verse is calculated. The maximum of these is

considered as the similarity score of the generated verse. A lower similarity score indicates a more novel verse.

A Good model should produce lyrics that have rhyme density similar to that of the artists and minimum similarity score.

4.3 Methodology

4.3.1 Approach1

Train the char level LSTM model directly on the fabolous data set. We noticed that the lyrics generated by this method had lot of grammatical and spelling errors. This can be due to the fact that the fabolous data set is too small for the model to accurately capture the grammatical and orthographic features of the language.

4.3.2 Approach2

- Pretrain the model on the kaggle data set to learn better character embeddings.
- Retrain the model (except the embeddings) on the fabolous data set.

The lyrics generated through this method had relatively less grammatical and spelling errors because better embeddings were learned using the larger kaggle data set capturing better orthographic and grammatical features

4.4 Evaluation

- Generate 10 verses from the model after every epoch during training.
- Calculate the average rhyme density of these generated lyrics.
- Save the model at the state where the rhyme density of the generated lyrics is closest to that of the artist.
- Calculate the similarity score of the lyrics generated by this model.

4.5 Results

Method	Maximum similarity
Baseline LSTM model	0.515
Character LSTM	**0.127**
Pretrained Character LSTM	**0.140**

The results show that character LSTM achieve lower maximum similarity scores at rhyme density 0.34. This can be attributed to the fact that character level LSTM can generate OOV words.

The similarity score obtained by the char level as compared to the pretrained model is low because the former had relatively higher number of spelling errors which lead to low similarity score.

A sample lyric verse generated by the model is shown below.

all they in the hand and do it like all dub with by this pill
and friends in the problems are on the razon just like me
im there for that cash hadded i jusned a right
i aint come out bikin as i wont see on a slack by a pring dough
thatk these feeling in the aught so i don t care for the yan
im standin a little bit on a chrome when i go and i aint wonderfly

5 Conclusion and Future Work

Through this paper, we have demonstrated the effectiveness of a character LSTM model for generating novel lyrics similar to that of an artist's. We have compared our approach with the baseline model and obtained lower similarity scores, signifying that the character level LSTM model is better at generating novel lyrics than the word LSTM language model.

For future work, we plan to incorporate constraints in the loss function that would help us model the style of the artist better. Further we intend to reduce the spelling errors produced by the character language model so that generated verses are more grammatical. Also other metrics for artist similarity could be used that can be directly incorporated in the training method.

References

Gabriele Barbieri, François Pachet, Pierre Roy, and Mirko Degli Esposti. 2012. Markov constraints for generating lyrics with style.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Hussein Hirjee and Daniel Brown. 2010. Using automated rhyme detection to characterize rhyming style in rap music.

Eric Malmi, Pyry Takala, Hannu Toivonen, Tapani Raiko, and Aristides Gionis. 2016. Dopelearning: A computational approach to rap lyrics generation. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 195–204. ACM.

Peter Potash, Alexey Romanov, and Anna Rumshisky. 2015. Ghostwriter: using an lstm for automatic rap lyric generation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1919–1924.

Ilya Sutskever, James Martens, and Geoffrey E Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1017–1024.

Dekai Wu, Karteek Addanki, Markus Saers, and Meriem Beloucif. 2013. Learning to freestyle: Hip hop challenge-response induction via transduction rule segmentation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 102–112.