DeepFire - Using Machine Learning Techniques to Synthesize Freestyle Rap

Justin Dieter - jdieter@stanford.edu, Alp Ozturk - aozturk@stanford.edu, Jonathan Zwiebel - jzwiebel@stanford.edu

Abstract

We present a generative model to synthesize a complete rap track given only a backing track. We isolated the task of generated lyrics and beats and solved both with Long Short Term Memory networks. Our model produced musically sound beats but failed to produce sensible lyrics. The synthesis of the two into a single track created musically valid by lyrically illogical rap.

Introduction

The production of hip hop music requires both a linguistic and musical sense of creativity. Developing an application to synthesize the lyrics, rhythm, and audio of a rap track represents a challenging machine learning problem. Through DeepFire, we utilized techniques that can be transferable to more widely applicable areas of both musical and linguistic analysis and synthesis and should provide a basepoint to make more progress in these areas. An overview of our algorithm is as follows. An input to our algorithm is any backing track that we desire to synthesize a rap track for. We used an LSTM network to generate both lyrics as well as rhythms. Once we have lyrics and rhythms generated, we synthesize audio using a text to speech library and processing the audio to make the words line up exactly with the generated rhythm. Finally, we overlay this synthesized vocal track with the original audio track.

Related Work

There have been few attempts to develop an algorithm for synthesizing a complete rap track including the lyrics and audio. However, there have been many projects analyzing the lyrics of songs and many projects analyzing musical features. We reviewed a project using an LSTM to synthesize drum tracks for Metallica songs, a project that generated rap lyrics, a project that generated Shakespeare-like poetry using an LSTM, and several others.

Dataset and Features

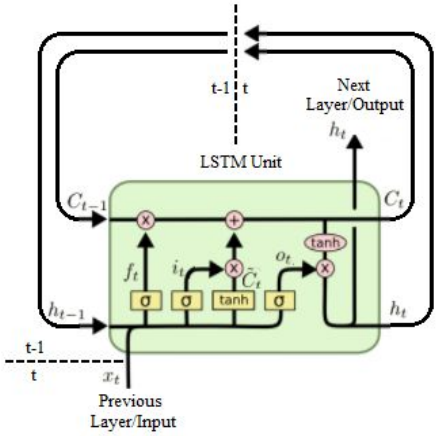
We used two models independently to generate lyrics as and rhythm. While there is definitely correlation between the two analyzing this was beyond the scope of our project. So, to train our lyric generation model we used a dataset consisting of billboard top 10 hip hop songs as well as a hand picked selection of some of our favorite songs spanning both older as well as

more recent rap songs. Our features for this particular model are simply the words considered as one-hot vectors in a vocabulary of all possible words. To train our rhythm generation model, we used the vocal track of a single rap song: “Fast Life” by Kool G Rap and Nas.

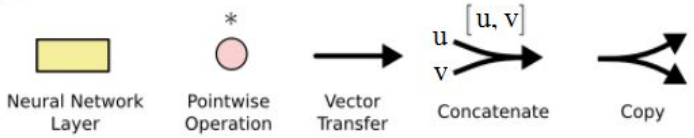
Methods

LSTM:

Both our rhythmic and lyrical generation used an LSTM (Long short term memory) architecture An LSTM network is a type of recurrent neural network that features a forget gate to allow the network to be able to learn what is important from previous data. In the case of lyrics this allows the model to have context rather than generating based solely on the previous words. In the case of rhythm this allows the model to understand a rhythmic pattern. The following diagram outlines the structure and update steps governing an LSTM network (Picture from http://colah.github.io/posts/2015-08-Understanding-LSTMs)/.



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$
$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$



In both of our usages, the input vector is a one-hot vector corresponding to words in a vocabulary and the output is a softmax. In this case, LSTM’s can be trained using gradient descent on a sequence based cross entropy loss function defined as follows:

$$l(\{y^{(t)}, p^{(t)}\}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_j^{(i)} log(p_j^{(i)})$$

Where $\{y^{(t)}\}$ is the actual sequence of words in the training data and $\{p^{(t)}\}$ is the predicted sequence of softmax values.

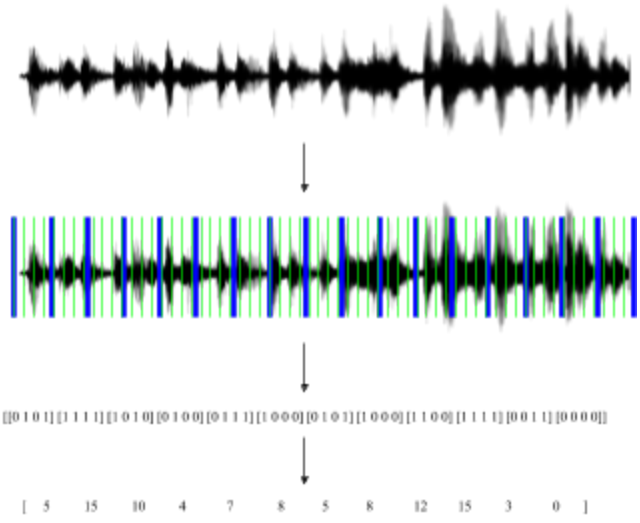
Lyrics:

Markov Model (Baseline): To get a baseline of performance we developed a naive markov process to create chains of length of 2, 3, and 4 of words in the input dataset. Output lines were generated by finding words that followed sequences of n - 1 words in our input set. For longer chain lengths we found that our output data too closely matched out input data.

LSTM Model: For the LSTM our features were simply sequences of one-hot vectors consisting of the words in the song viewed as elements of a vocabulary containing all words used. We chose to use a training rate of 0.003 with 32 units in our model and we tried both a 1 and 2 layer model to compare the performance of both. These parameters were chosen because they seemed to provide a sufficiently large model to be able to learn patterns in the data and be reasonable to compute. Also the training rate was selected because our model had no problem converging with it.

Rhythm:

In order to make our LSTM model for rhythm generation we first had to decompose the rhythm of a track into analyzable features. We used LibROSA to determine the tempo of an audio sample as well as the timestamps of beats and musical onsets. In our case, the musical onsets corresponded to the start of vocal utterances in the rap track (In the diagram below blue lines correspond to beats and green lines correspond to musical onsets. Note that the diagram is not accurate and is only for purposes of illustration). Since hip hop music has very little triplet-like rhythms, it seemed to be a valid approach to round the locations of each musical onset to within a sixteenth note. This allowed us to produce a sequence of binary values for whether a vocal occurrence occurs in each possible sixteenth note duration (Third step in the diagram). Last we process each beat (4 binary values in the sequence) to be considered as a word in a vocabulary of 2^4 distinct possible rhythms. From here we were able to convert to a sequence of one-hot vectors and use an LSTM model in the way described earlier.



Beat Vectorization

Synthesis:

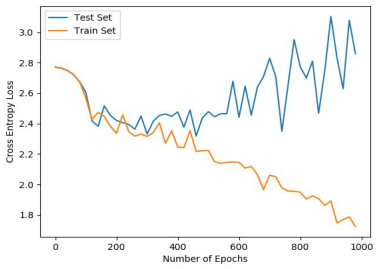
In order to synthesize lyrics and beat we used the text to speech API Amazon Polly to get audio files for each of the words in our vocabulary. We used the textstat library to count the number of syllables in each word of our generated lyrics. We then used the Rubber Band library to shrink the audio for each word to only take up as many musical onsets as it contains syllables. Then, we overlaid this generated audio with the original backing track to get our final output.

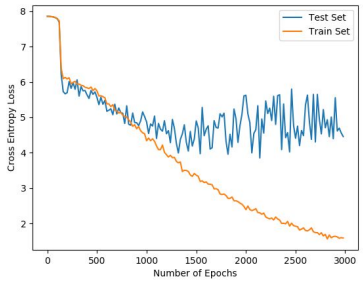
Results

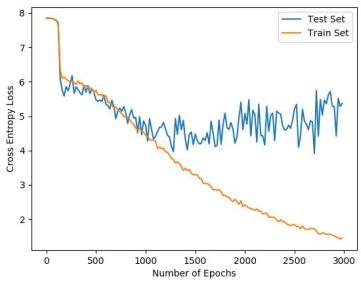
We have included both numerical analyses and links to the tracks produced by our model for both quantitative and qualitative analysis.

Training:

Below is a plot of the cross entropy loss on the test and train sets for both the rhythm generation LSTM (left), lyric generation LSTM with a single layer (middle), and lyric generation with two layers (right):







These results show that none of our models generalized especially well and are subject to much overfitting. Due to the small size of our training data it is likely that a much bigger training dataset is necessary to obtain better generalization for our model.

Quantitative Analysis:

For quantitative analysis we created a script to analyze common patterns of parts of speech and to count their relative proportions over a text. We then ran this script over the raw rap input data, the output of our Markov model, and the output of the LSTM model to determine the grammatical similarity between our different tracks.

`adj-noun` tests the proportion of times a noun correctly follows an adjective
`noun-verb` tests the proportion of times a verb correctly follows a noun
`adverb/verb` tests the proportion of times that an adverb is correctly next to a verb

`noun-verb-noun` tests the proportion of times that after a noun-verb construction, a noun phrase is correctly used
`squared-difference` is the difference between this dataset and the rap.txt dataset

	adj-noun	noun-verb	adverb/verb	noun-verb-noun	sum- squared difference
rap.txt (test data)	0.880	0.482	0.582	0.935	N/A
Markov 2-chain	0.585	0.358	0.342	0.639	0.248
Markov 3-chain	0.635	0.470	0.333	0.771	0.149
LSTM 1 layer	0.692	0.358	0.452	0.636	0.157
LSTM 2 layer	0.580	0.516	0.411	0.740	0.158

This demonstrates similar performance between our 1 and 2 layer LSTMs at matching the grammatical structure of the input rap data. Both LSTMs performed similarly well to the Markov 3-chain model and significantly better than the Markov 2-chain model.

Here is a link to a single track produced by our model:
https://soundcloud.com/justin-dieter-927098414/deepfire-rap

Future Work

Our model clearly has a long way to go to achieve anything near human level performance. Possible improvements could be any of the following:

- Training lyric generation on a much larger training set. http://www.ohhla.com/ contains the lyrics to hundreds of songs. Writing a web crawler to extract data from here could produce a significantly better corpus of lyrics.
- Designing a model and collecting training data to consider the correlation between rhythm and lyrics in a hip hop song
- Utilizing state of the art NLP techniques to get a better model for lyrics generation rather than a simple LSTM.
- Designing a custom text to speech engine to have inflections and tone in a similar way an actual rapper would.

References

[1] Malmi, E., Takala, P., Toivonen, H., Raiko, T., & Gionis, A. (2016). DopeLearning. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD 16. doi:10.1145/2939672.2939679

[2] ArXiv, E. T. (2015, May 20). Machine-Learning Algorithm Mines Rap Lyrics, Then Writes Its Own. Retrieved November 21, 2017, from https://www.technologyreview.com/s/537716/machine-learning-algorithm-mines-rap-lyrics-then-writes-its-own/

[3] Cooper, M. (2017). Encore.ai [Computer software]. Retrieved from https://github.com/dyelax/encore.ai

LSTMetallica: Generation drum tracks by learning the drum tracks of 60 Metallica songs. (2016, July 16). Retrieved November 21, 2017, from https://keunwoochoi.wordpress.com/2016/02/23/lstmetallica/

[4] LibROSA[Computer Software]. (2017). Retrieved from https://librosa.github.io/librosa/index.html

[5] Rap Music: Top Rap Songs Chart. (n.d.). Retrieved December 12, 2017, from https://www.billboard.com/charts/rap-song/2017-12-09

[6] The Best Rap Songs of All Time. (n.d.). Retrieved December 12, 2017, from https://www.ranker.com/list/best-rap-songs-of-all-time/ranker-hip-hop

[7] spiglerg (2017) Text generation using a RNN [Computer software]. Retrieved from https://github.com/spiglerg/RNN_Text_Generation_Tensorflow