CS230

Generative Lyric Composition using Transfer Learning

Abraham L. Apellanes
Department of Computer Science
Stanford University
aapellan@stanford.edu

Jake Wagner
Department of Computer Science
Stanford University
jtwagner@stanford.edu

Abstract

Our project explores modern deep learning techniques and the potential they have to supplement the songwriting process by learning from artists and their extensive work. We build on a neural network pre-trained on Kanye West's discography by tuning the model's hyper parameters with our own dataset of 4,000 songs from prominent hip-hop artists and rappers from the past 25 years. We found that with the right hyperparameter tuning our model yields 5% loss. Qualitatively, we found that people could still reasonably discern Soundcloud rap lyrics from our models generated lyrics. However, we did find that some of the bars of lines of lyrics produced were very high quality. Overall, our results show promise, indicating that deep learning techniques can reasonably be used to supplement the creative process of songwriting.

1 Introduction

We present a generative model that utilizes a recurrent neural network (RNN) and transfer learning to augment the songwriting process. Artists spend their careers honing their abilities to construct expressive, relatable, and creative lyrics. We saw this as a task that could be supplemented with artificial intelligence. Instead of having struggling artists work through hours of possible content, what if we could augment the process and have a neural network generate line after line of reasonable content that the artist can then pick and choose from.

We thought this problem held significance because finding reasonable lyrics that follow a strong rhyme scheme are hard to develop, but with artificial intelligence we can not only make that process faster but also easier for artists. We thought it would be super interesting to see if our neural network could produce ten bars of lyrics that were indistinguishable from SoundCloud rap and hip hop artists of today.

The input to our algorithm is a Kaggle data set of 4,000 hip hop songs from thirty five top hip hop artists of the last twenty five years. These artists include Eminem, 2 Pac, Drake, 2 Chainz, Busta Rhymes, etc. We then use a neural network of LSTM layers and a rhyme index to output a full set of lyrics. This model is generative and with each trial we ran, we got a different set of lyrics.

2 Related work

2.1 Foundational Literature on RNNs and LSTMs

Sutskever et. al detail many of the current problems regarding RNNs, while also highlighting their recent improvements as stepping stones to a less expensive model that more accurately depicts global features and reflects thematic details in text (Sutskever). Sutskever et. al introduce RNNs as powerful sequence models with existing training difficulties. One such difficulty researchers have encountered is the vanishing/exploding gradients problem (Sutskever). Recent advancements have led to the Hessian-Free Optimizer which significantly reduces the gradient descent problem and makes training RNNs much more stable. Sutskever et. al find that by using RNNs and a Hessian-Free Optimizer as components of a much larger model that they could outperform industry benchmarks.

Graves et. al explore text generation using LSTMs; describing the basic RNN and LSTM architecture, defining the purpose of a recurrent neural network where the RNN makes weighted has a robustly connected layer where the input weights affect each hidden layer which then feed forward into the final output layer. The output of one layer is used to make a prediction for the input of the next layer. As a result, the RNN learns from its previous predictions. It is of importance to note that Graves et. al add stacked LSTM layers to their model. One major problem with RNNs is that they fail to "remember" information produced a variable number of sentences prior, or in some cases even some words prior. By using the stacked LSTM layers, Graves et. al hoped that their model would be able to recollect features, but they instead found that with larger networks and numerous iterations RNNs overfit to their data. With this insight in mind, Graves et. al applied regularization techniques, which achieved positive results.

Sutskever et. al and Graves et. al describe the RNN and LSTM frameworks in depth. They are paramount for understanding the basics of this model. Our review of existing literature becomes more focused on generative models, beginning with Ghazvininejad et. al. and their neural net application in poetry generation.

2.2 Downstream Generative Tasks

Ghazvininejad et. al investigates the use of RNNs for the production of topical sonnets (Ghazvininejad). Their insights on post-process and guidance of their RNN differed significantly from most existing work we explored. Ghazvininejad et. al developed a standard RNN save the addition of a vocabulary-based framework for a rhyme scheme associated with a topic or theme which they wanted to generate a poem about. For instance, when creating a poem about sunshine, the model would construct a rhyme scheme with words like "yellow", "warm", "bright", and so on ensure the poem's content is consistent with the theme. Ghazvininejad et. al also perform syntactic post processing of items such as "a" vs. "an" to make it appear more realistic. These techniques made their RNNs more robust, making it appear as if a human manually aided their performance on testing. We felt that this took away from the architecture because it forces a continuation of the topic and grammatical validity instead of having the model learn grammatical truths.

Chu et. al take a creative approach to creative song generation using a song based on the A-minor harmonic scale and subsets of digits from the number pi (Chu). Instead of looking solely at textual evidence and producing lyrics, Chu et. al used different layers of a hierarchical RNN to produce different features of the song. They had layers for the key, chord, press, and drum. They explain that any conceivable sub-sequence of digits can be formed from pi, and that this randomness, interestingly enough, can be harnessed to compose a song that outperforms model performance standards set by recent generative projects, namely the Google Magenta Project (Elliot Waite). We derived insight about the importance of randomness subjectivity in Chu et. al., especially given the nature of our own project.

Potash et. al use an LSTM RNN architecture to create rap lyrics without any guidance in terms of post processing for rhyme schemes or max syllables per line (Potash). Potash et. al decided to test the strength of these neural networks by placing no processing limitations after the lyrics had been produced. Potash only used lyrics from one artist (Fabolous) and compared their model to that of an n-gram model. Using only one artist proved effective for them because they were better able to model the style of the artist while also exhibiting a better rhyme scheme than the n-grams model.

Their model was more dynamic in it's production. The concern we had about Potash et. al's approach is that there exists more robust models to compare.

Existing literature showcases the application of the LSTM RNN architecture for various artistic modelling tasks. The creation of sequential works is shown by the development of melodies as well as language models. LSTMs and RNNs have been proven as promising and novel frameworks for downstream generation tasks in deep learning.

3 Dataset and Features

Our dataset consists of approximately four thousand songs from Metrolyrics.com which is availableon Kaggle. We did not divide our dataset into training, validation, and test sets because our task is generative. Instead, we used all four thousand lyrics for training and developing lyrics. Each entry contained a song's title, year of release, artist, genre, and lyrics. This data came in a .csv file with the following columns: [index, song, year of release, artist, genre, lyrics]. We culled our list to thirty-five prominent artists then retrieved all of their song lyrics from our dataset. Our preprocessing overhead was large. We razed entries with inconsistencies, errors, and unnecessary text such as "[Verse 1]" or "[X3]". or errors.

We put the clean data into a matrix with only lyrics. From there we pushed all of the lyrics into a text file that could be read into the model. We performed some data normalization, splitting the lyrics based on new lines. The only features that we used in generating lyrics were the lyrics from our dataset. In this LSTM RNN architecture, it didn't include any possible input for other features. Instead, it relied on the ordering of the dataset and relations between the features (text) that are apparent in the dataset itself. Another extracted set of features that we obtained was the set of weights that Barrat initially trained his model on. These weights were instrumental in applying transfer learning.

4 Methods

We use transfer learning on an LSTM RNN. Our initial model is trained on the discography of Kanye West–approximately 200 songs–and comes from Robbie Barrat (Barrat). We adapted it to take a data set consisting of 4000 songs from 35 artists different artists, observing the loss and the quality of the model's lyrics. The RNN relies on the recurrence of previous data and layers in order to make the current predictions. The mathematical function for the RNN is as follows:

$$a^{<t>} = g(W_{aa} * a^{<t-1>} + W_{ax} * x^{<t>} + b_a$$
$$\hat{y}^{<t>} = g(W_{ya} * a^{<t>} + b_y)$$

(Ng) "Recurrent Neural Network Model"

Here the "x"s act as inputs from the different layers, the $\hat{y}$ acts as the prediction for that layer, the "a"s act as the activations. "g" represents different activation functions. The "W's represent the weight matrices with subscripts showing which point in the hidden layer that they exist.
This RNN uses the LSTM architecture to produce these weights and make its predictions. The math is much more extensive for the LSTM units but appears as the following retrieved from Andrew Ng's slides on LSTMs:

$$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c)$$
$$\Gamma_u = \sigma(W_u[\, a^{<t-1>}, x^{<t>}] + b_u)$$
$$\Gamma_f = \sigma(W_f[\, a^{<t-1>}, x^{<t>}] + b_f)$$
$$\Gamma_o = \sigma(W_o[\, a^{<t-1>}, x^{<t>}] + b_o)$$
$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>}$$
$$a^{<t>} = \Gamma_o * \tanh c^{<t>}$$

(Ng) "LSTM (long short term memory) unit"

In the above formulas, the top equation of c represents the candidate value which takes in the current weights, the previous activation, the current input, and a bias. These variables are similar to that of the RNN structure in what they represent. "W"s represent weight matrices at different times. Γ represent the gate functions for the LSTM for the topics to be remembered or forgotten. "b" represents the biases. "c"s represent candidate values, and "a"s represent particular activations.

This combination of LSTM units making up the RNN form much of the mathematical foundations. One other architecture decision that was made by Robbie Barrat for this LSTM RNN that we used was the mean squared error loss function. MSE takes the square of the loss at every iteration in the network. A facet of this neural net which differentiates it from some works is its use of a rhyme index to produce lyrics; it indexes all of the suffixes of the end words in the training data. During post processing, it compares the proposed line with its most recently generated line to make sure that the rhyme scheme matches. This is important to our methodology because it forces a rhyme scheme and makes the lyrics appear more authentic. This can be harmful to the coherence of lyrics between lines because it is forcing a specific rhyming format and limits the number of possibilities for production.

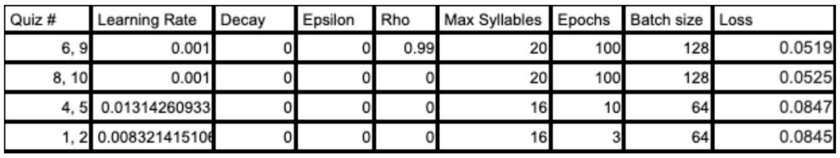
Now in using transfer learning we utilized the weights that the model had learned on Kanye West's discography and applied that to a much larger dataset. Once we had finished this part of the process we began hyperparameter tuning using the panda process. Andrew Ng describes the panda process as a procedure to minimize computational expense and optimize time for performance. The fault of this process comes from the fact that you don't get to evaluate as many hyperparameter tunings and must believe that the developer has selected the best hyperparameters to tune. The results of the experimentation and the optimal hyperparameters to tune will be described in the following sections.

5 Experiments / Results / Discussion

We performed trasnfer learning, tuning the learning rate, learning rate decay, epsilon, Rho, maximum number of syllables per line, number of epochs during training, and batch size. Initially, we tried to work with the default batch size of four, but this had a similar effect of stochastic gradient descent where the gradient descent of the loss function was extremely noisy. We found that increasing the batch size to 64 and 128 units made for a stabler gradient descent and decreased the training time significantly. We further tried to combat the noisy gradient descent by tuning Rho, hoping that some momentum may speed our gradient descent and make our resulting loss more accurate. We increased our number of epochs from 3 to figures up to and between 10 and 100. Consequently, we encountered a vanishing gradient problem. However, we eventually realized that this was more likely true convergence rather than a vanishing gradient.

Through the majority of our hyperparameter tuning, our model plateaud with a loss of 8%. We attributed this to the lack of our models ability to adapt to stark contrasts between subsets of the lyrics from a large variety of artists. We reached a breakthrough when tuning the maximum number of syllables allowed per line from 16 to 20. Our loss dropped from 8% to 5%. We conjecture that this tune would allow the model to have more freedom in creation, allowing for more words to be inserted before the end rhyme needed to be made. This reduction brought us much closer to the initial model's loss of 4%, which Barrat achieved training it solely on Kanye West's discography.

Once we had a sufficient number of lyrics from various tunings, we randomly sampled from our resulting productions and selected two sets of lyrics with a 5% loss and two other sets of lyrics with an 8% loss. Once we had our four sets of lyrics, we randomly sampled ten lines from each.

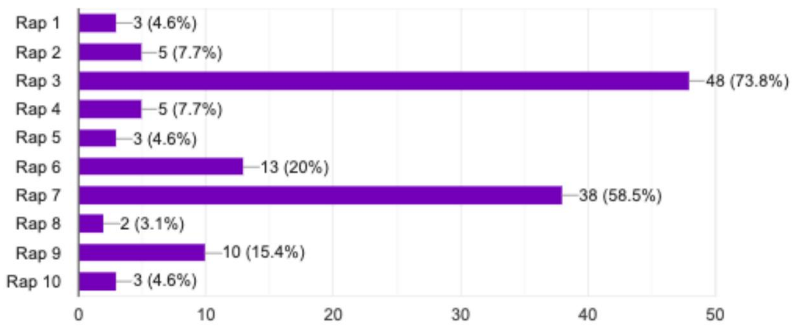


Quiz #	Learning Rate	Decay	Epsilon	Rho	Max Syllables	Epochs	Batch size	Loss
6, 9	0.001	0	0	0.99	20	100	128	0.0519
8, 10	0.001	0	0	0	20	100	128	0.0525
4, 5	0.01314260933	0	0	0	16	10	64	0.0847
1, 2	0.008321415106	0	0	0	16	3	64	0.0845

We then selected two rap songs from trending SoundCloud artists to form a total of ten lyric samples of ten lines each. We used SoundCloud because recently trends in rap have proven SoundCloud to be a viable platform for an artists early career and because artists on SoundCloud are less well known.

We created a Google form with the ten samples of lyrics and began asking people to rate each of the ten samples of lyrics and then try to decipher which of the two samples were from human SoundCloud artists. The following paragraph gives our results from this survey.

After surveying sixty five people we found that 73.8% of individuals surveyed were able to identify Jay Critch's song "Nintendo" (Critch) while only 58.5% of people were correctly able to identify "Fa Sho" by Devin the Dude (Devin the Dude). The top two false selections were samples from the lowest loss generations of lyrics. The sixth generation performed the best out of the artificially produced lyrics, having been chosen 20% of the time.



On this graph, the y-axis represent each of the different rap songs 1-10, where 3 and 7 are the Soundcloud rap lyrics and the rest are all generated by our network. The hyperparameters for each of the raps are listed in the previous table. The x-axis shows the percentage of people that selected that rap as a true Soundcloud rap. This graph emphasizes the potential of LSTM RNN models for creative generation. Another note on the experimentation of our project is that we only took snippets of ten lines, and although done randomly, it is more difficult to determine themes which would be much more apparent in a full length song. This gave aid to our testing because we didn't need to have a completely cohesive piece, but instead could rely on closer connections and metaphors made in a shorter sequential time span. Our results indicate to us that AI and deep learning will not overtake the current songwriting process anytime soon.

6 Conclusion

A qualitative test of our model's performance showed participants could not discern our model's lyrics from the lyrics of trending SoundCloud rappers up to 20% of the time. The survey results and the feedback we received indicate that while our model can produce bars of coherent lyrics, it is far from comparable to lyrics which are human-composed. As expected, our model lacks the ability to produce large sets of lyrics which exhibit real thematic structure and a varied, creative flow. This lyric generation could instead be used to help artists in the songwriting process to get ideas and move away from writer's block. In terms of future work, we would have liked to utilize more compute power to take a Fish Approach to hyperparameter tuning. With this approach, we could more fully experiment with the loss function when changing things such as average line length of the train data and the maximum syllables per line. Finally, we would have liked to conduct experiments to test our results in a more controlled manner and increase user trials to glean more insight as to what qualitative aspects our model we should seek to achieve to be able to produce better lyrics.

Authorship / Contribution

We co-authored the written reports for the project proposal, milestone chec-in, and final paper. Jake took the lead on the development of the initial code needed for data scraping and pre-processing. Abraham researched numerous papers to glean applicable approaches and ideas. Together, we discussed the performance of our models, tuned hyperparameters, and performed final tests and qualitative and quantitative error analysis.

Code

To view the code for the neural network, the link to the Google Colab notebook is: https://colab.research.google.com/drive/1saNGO8QXrEE-c7pEd5-07CsrRMOXepkn
The view our preprocessing code, the link to the github repository is: https://github.com/jacobwagner8/LyricGenerationNN

References

Barrat, Robbie. "Rapping-neural-network". https://github.com/robbiebarrat/rapping-neural-network

Chollet, Fran. "Keras." 2015. https://keras.io

Chu, Hang, Raquel Urtasun, and Sanja Fidler. "Song from pi: A musically plausible network for pop music generation." arXiv preprint arXiv:1611.03477 (2016).

Critch Jay, Rich The Kid. "Nintendo." https://genius.com/Rich-forever-music-nintendo-lyrics. 2017.

Devin the Dude, Odd Squad. "Fa Sho." https://genius.com/Devin-the-dude-fa-sho-lyrics. 2002.

Elliot Waite, Douglas Eck, Adam Roberts, and Dan Abolafia. Project magenta. https: //magenta.tensorflow.org/.

Ghazvininejad, Marjan, et al. "Generating topical poetry." Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. 2016.

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

Ng, Andrew. Course 5 Module 1 Slides.

Oliveira, Hugo Gonçalo. "A survey on intelligent poetry generation: Languages, features, techniques, reutilisation and evaluation." Proceedings of the 10th International Conference on Natural Language Generation. 2017.

Potash, Peter, Alexey Romanov, and Anna Rumshisky. "Ghostwriter: Using an lstm for automatic rap lyric generation." Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. 2015.

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