Lyrics Generation: NLP Project for ENSF 519

Jenard Cabia, Shuyi Jin, Nathan Schuetz, Quan Sun

***Abstract*— an NLP project is done to examine the capabilities of machine learning for exploring existing song lyrics, and for generating new song lyrics after being provided sentences to start with. A set of song lyrics and their metadata was scraped from two websites. This data was explored in detail with the help of Natural Language Processing techniques(e.g. Sentimental analysis) and then used to train a LSTM neural network to be able to produce new lyrics data. Two models(character-wise model and word-wise model) are utilized and a comparison study is done. The results show that word-wise model performs better than character-wise model and both can generate lyrics which make grammatical sense after training for a certain amount of epochs.**

I. INTRODUCTION

Artificial Intelligence which was considered a topic of the future back in the 20th century, is now an emerging topic in the field of Computer Science and has been on the rise especially over the past decade, contributing a huge portion to the advances made in technology. In this paper, we will explore a few subtopics within the domain of Artificial Intelligence that have garnered a lot of attention and research over the past couple of years which are Natural Language Processing, Natural Language Generation, and Machine Learning Techniques such as Deep Learning.

II. BACKGROUND

A. Problem

The topic of interest to be discussed in this paper is whether Artificial Intelligence can be trained and optimized well enough to generate sequences of meaningful song lyrics for various genres and moods. Proving that AI can express the same thoughts and feelings that song writers had when writing lyrics in the past and the present will not only be proof of significant advancements in the field of NLP, it will also enable preservation of song lyrics from previous generations which will influence future song lyrics written by AI in the years to come.

B. Solution

In order to assess the natural language generation capabilities of Artificial Intelligence, we began our research by exploring the problem domain and related topics towards automated sequence generation of song lyrics. One vastly popular approach of sequence generation we found in our study which we will utilize in this experiment is an Artificial Neural Network (ANN) which will be trained through supervised learning using song lyrics labeled per genre and mood. Since lyrics are sequential data and we want the network to remember parts of the sequences from previous iterations, we will also implement a deep learning model Long short-term memory (LSTM) to satisfy our requirements for sequence generation.

Our main approach was to divide the project into two phases, Phase 1 & 2. Phase 1 will focus on collecting the information and data required for lyrics generation through scraping of multiple sites including allmusic.com and lyrics.com. Each song collected will be cleaned, integrated and labeled before it enters the preprocessing stage where a text-normalizer is used to remove html tags, accented characters, special characters, and stop words. Stemming/Lemmatization will also be performed to the full set of lyrics which will be our corpus, making a document to be the lyrics of a single song. Songs will be labeled as either positive or negative by performing Sentiment Analysis on each document using VADER Lexicon. Once labeled, the data will be stored in a Pandas DataFrame where it can be classified and grouped by genre, by artist, and by album. Lastly, the sentiment results, as well as common word occurrences for each grouping will be visualized using MatPlotLib.

Phase 2 will focus mainly on designing the ANN and LSTM models required for lyrics generation, reporting and analyzing the newly generated song lyrics, and lastly performing an evaluation of the results to see whether the phrases used and the emotions of the generated lyrics reflect that of the training set. The data gathered from Phase 1 will be used as training data for the models as well as the expected outcome for the algorithm.

III. EXPERIMENT

A. Objectives

- Explore the problem domain and cover the required background
- Collect song information such as lyrics, genre, artist, year, album, mood, and any other information through the means of web scraping to populate our dataset
- Integrate scraping results and exclude any entries with missing critical information
- Normalize the full set of lyrics collected by removing html tags, accented characters, special characters, stop words, and perform stemming/lemmatization
- Apply sentiment analysis using VADER Lexicon to label each song either as positive or negative
- Generate a lyrics corpus that will be used to generate new lyrics
- Perform exploratory analysis and visualization for various groupings of the dataset such as by artist, by genre, and by album
- Create the LSTM character-wise and word-wise models using Python and TensorFlow (Keras)
- Report the results of the analysis through various visualizations, statistics, and measurements
- Validate the results obtained, perform model evaluation, and display the results through figures/tables
- Validate the results by creating actual songs with music as proof of concept

B. Procedure

1) Data collection and integration:

What was desired was a large corpus of song lyrics along with the metadata needed to analyze it using statistical tools and supervised machine learning techniques. In particular, it was valuable to obtain mood labels for each song. To this end, allmusic.com was selected to provide both song lists and metadata, and lyrics.com was selected to provide the lyrics themselves.

Because allmusic.com categorizes music by mood, a web scraping program was used to obtain links to each mood's music list. Each of these mood pages was then scraped to obtain a list of songs corresponding to that mood. A difficulty arose in that the individual songs are not often labeled with a mood, and after further exploration of Spotify, Google, Apple, and YouTube playlists a reliable source of labels was not found. YouTube in particular had many partially labeled playlists, but their contents were almost always divergent from the labels and the metadata was formatted so inconsistently that addressing it could have been a project in its own right. One possible solution was to still make use of the mood pages of allmusic.com, but to obtain lists of mood-labeled albums rather than individual songs. The album's labels could then be applied to the individual songs. This was not ideal because almost no albums have songs with only a single 'mood.' For example, many of the saddest albums contain at least one happy song. Because a guideline in the keras documentation suggested that a LSTM model may require datasets exceeding one million characters for character-wise models.

In order to increase the size of our dataset by ten-fold, the compromise actually settled upon was to use the results of a subsequent sentiment analysis step to organize the songs prior to training a model, and to modify the original objective slightly due to this.

After obtaining a list of songs, lyrics.com was scraped using its search feature to obtain lyrics for each song. If lyrics were not found for a given song, that song was simply omitted from the corpus. Finally, if multiple songs by the same artist shared a name, it was assumed, with some informal verification, that the different versions were either re-releases of one another, or else similar enough in content to not contradict our mood or genre labels.

2) Preparation:

The text data scraped from the lyrics website needs to be cleaned or preprocessed before they can be further analyzed or fed into the training models to get a more accurate and meaningful result [1].

The lyrics are normalized in the following steps:

- Remove HTML tags
- Remove accent characters
- Remove duplicates
- Remove extra newlines and extra white spaces
- Remove special characters
- Expand contractions
- Lemmatize the words
- Remove stopwords

Finally, a normalization function combining all the functions above is created.

Note that the negation words (no and not) are kept when removing the stopwords from the text data, since they might be useful during sentiment analysis. Another version of the preprocessed dataset has kept all the stopwords. This dataset is going to be trained in Phase 2 to generate new lyrics.

In order to implement all the preprocessing functions, spacy and nltk are needed, both of which are state-of-the-art libraries for NLP. In addition, contractions.py module [2] is required to download for the purpose of expanding contraction.

In the end of the preprocessing step, the original DataFrame is inserted with two new columns: normalized lyrics with and without stopwords.

The following is an example of lyrics preprocessing:

- Lyrics(original):I was blind, now I can see You made a believer out of me I was blind, now I can see You made a believer out of me I'm movin' on up now Getting out of the darkness My light shines on My light shines on My light shines on I was lost Now I'm found
- Lyrics(normalized_no_stopwords):blind see make believer blind see make believer movin get darkness light shine light shine light shine lose find
- Lyrics(normalized_with_stopwords):i be blind now i can see you make a believer out of me i be blind now i can see you make a believer out of me i be movin on up now get out of the darkness my light shine on my light shine on my light shine on i be lose now i be find

3) Exploration:

Lyrics as a form of art, can bring people with joy, sadness and all other kinds of emotions. To analyze them, it is necessary to know the emotion the lyrics trying to express. Through sentiment analysis, the song lyrics can be classified as positive, negative or neutral.

All the song lyrics are first grouped by the mood of their album, then analysed to find the most positive and most negative song belongs to each mood. For the songs with extreme scores are found and analyzed the word components and frequencies they appear.

The songs are then grouped by their genres and analyzed to find the sentiment scores of songs that fall in the same genre. For each genre, the top 50 words with the highest frequency are visualized.

Finally, the songs are grouped by the artists, and sentiment analysis is applied to see how their songs vary in sentiment scores throughout their albums over time.

4) Prediction:

Our objective is to build a deep learning model that can take a sequence of words as an input and output a continuation of that sequence, i.e. generate new lyrics with regard to our project. For example, if we give our model the string "I love Canada", we expect lyrics for a song starting with "I love Canada" as the output.

To achieve the goal, we need to build a suitable Artificial Neural Network (ANN) model. Since text data is sequential data and we want our network to remember parts of the sequence it has seen many time steps back, our model

has to have a short-term memory and a long-term memory. Therefore, we will design/implement a deep learning model Long short-term memory (LSTM) which satisfies our requirements and conducts sequence generation.[9]

Generally, we will follow the procedures as follows:

- Firstly, take the data (lyrics) by web scraping as the training data;
- Then, code the LSTM model using Python and Tensorflow (Keras);
- Thirdly, according to given corpus, train the model using supervised learning where the network is going to try to fit an array of sequences to an array of characters/words both of which are generated from the training data;
- Finally, we input the string we want our lyrics to start with and the model generates new lyrics.

In order to train such a model efficiently, it is typical to use one or more GPUs to allow the task to be parallelized. Two approaches to this were taken. The primary method used was to take advantage of Google Colab, a free service where jupyter notebooks can be hosted and run with the use of a Tesla k80. Due to this surprising opportunity, it was possible to train a LSTM model very early without significant configuration. However, since these training sessions of 100 epochs each took over 6 hours to perform, tensorflow and keras were additionally installed on a desktop computer that happened to have a gtx 1080 available. This second machine was used in parallel to test the effects of alternative model parameters.

C. *Measurements*

VADER Sentiment Analysis

- For sections of the project that utilized Sentiment Analysis, we used the metric 'compound score' as a means to determine the sentiment score of the text being examined. We chose this metric as it corresponds to the sum of all the lexicon ratings or the polarity scores(positivity, negativity, and neutralness) of the text being examined.

For LSTM models,

- our project is a classification problem, and the output layer has a certain number of units corresponding to characters or words which can be seen as categories or classes(later we will talk more about this), therefore we use categorical cross-entropy as loss metic for gradient descent.
- Categorical cross entropy is as follows:

$$CE = -log(S(yi)) \;\; i: the\ true\ label\ number$$

- The final loss is

$$L = \sum_{j=0}^{N} CE \,, N\ is\ batch\ size$$

- Categorical accuracy is also be used to measure how accuracy our models generate characters/words.

D. *New techniques from outside of class*

1) Sentiment analysis:

Sentiment analysis may be hard to perform, and even humans can get misled sometimes. Since a piece of text main contain multiple sentiment at once, and computers are not good at comprehending figurative language. In figurative language, words are used in a way that is different from their conventional definition to convey a more complicated meaning. A word can express a completely opposite emotion with its context in the sentences. As a result, sentiment analysis of text data can hardly ever to get 100% accuracy. These are some open-ended problem in the Natural Language Processing field.

The text data collected in previous steps are not well-labeled by their sentiment, so sentiment analysis has to be done using unsupervised lexicon-based approach. A lexicon is a special dictionary, vocabulary or a book of words created for analyzing sentiment. There are many popular lexicons, but most of them have a list of positive and negative polar words associated with scores. VADER (Valence Aware Dictionary and sEntiment Reasonor) lexicon is the sentiment analysis tool used in this project[3][4].

VADER lexicon works well on social media type of text, and it does not require any training data. To use the VADER lexicon, the SentimentIntensityAnalyser is imported and called, after installing VaderSentiment using pip. Next, polarity_score() method is called to obtain the polarity indices for the given text. The method outputs the positive, negative, neutral and compound score of the input text. The positive, negative and neutral score represent the proportion of text falling in these categories. The compound score is computed by summing the valence scores of each word in the lexicon, adjusted according to the rules, and then normalized to be between -1 (most extreme negative) and +1 (most extreme positive).

2) LSTM network:

There are two approaches used: character-wise model and word-wise model.

a) Char-wise model:

Firstly, the corpus(lyrics normalized with stopwords) is divided into sequences as the input xs and the length of each sequence for our model is 40. Meanwhile, the skip step/stride has to be determined, 3 for our model. So, the first sequence is from character 0 to character 40 and the second sequence is from character 3 to character 43, until the end of the corpus. By the way, the length of the final sequence might be less than 40.[7]

Corresponding to each sequence, the character in the corpus just next to the sequence will be selected as y_label. The number of y_labels is the same with the number of sequences.

As for the input sequences and y_labels, they are strings and characters which cannot be dealt by computer and they have to be transformed into numeric numbers/vectors. A vocabulary is created containing all the unique letters a-zA-Z and blank whitespace, therefore, 27 characters all together. Based on the position of each character in the vocabulary, all the characters in the input sequence and y_labels can be transformed into one-hot encoding vectors.

After training data xs and ground truth y_labels are ready/vectorized, Keras, a high-level deep learning framework is used to build the LSTM model. One layer LSTM with 64 neurons/nodes is utilized and shown in an unfolded form below. For this model, only the last LSTM

cell/block outputs the predicted h having the same length 64 with the hidden layer size.[8]

The layer before the output layer of this model is a dense/fully connected layer with 27 units corresponding to the number of characters in the vocabulary. And then softmax transform is applied to the outputs of dense layer which are called logits and generates probability distribution for each character. The character with the highest probability will be the predicted y which is the most suitable letter to append to the input sequence.

Note: the number 128 in the figure below means batch size, and softmax function is

$$S(yi) = exp(yi)/\sum_{j} exp(yj) \quad i = 0,1,..,26$$

where yi/yj is logit.

During training, after 128(batch size) sequences finish input and forward pass, outputs of each layer including the output layer are generated. Then, categorical cross-entropy will be used as the loss function and stochastic gradient descent with the momentum will be used as optimizer to back propagation using chain rule.

Through calculating the gradient of L and backpropagation, weights of each layer will be updated. Therefore, generally speaking, the model will trend to perform better with epochs increasing.

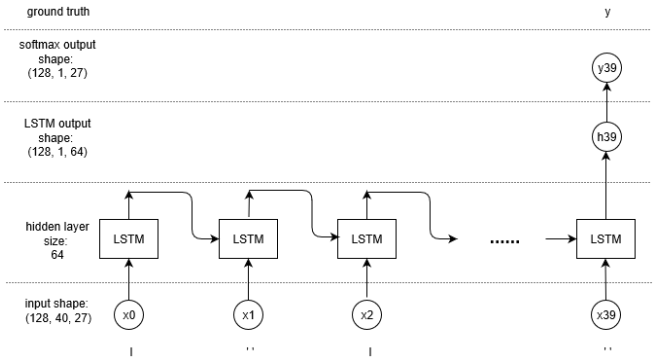


Figure 1 Character-wise model

b) Word-wise model:

Different from char-wise model, word-wise model inputs word. However, because the machine cannot handle words/strings, input words have to be transformed into a form of numbers. Here, word2vec/word embedding technique needs to be used[6]. Firstly, a vocabulary of all the unique words in the corpus is created, and then similar to char-wise model, each word will be transformed into one-hot encoding vectors. For our model, the length of the vocabulary is 5201 meaning our corpus has 5201 unique words. So, each word will be transformed into a 5201-length vector with only one 1 and the others 0. A softmax trainer for word embedding can be used to transform 5201-length vector into 300-length vector which is not one-hot vector. The dimensionality is reduced.

Similar to the process of generating input character sequences and y_labels of char-wise model, word-wise model also generates input word(vectorized) sequences and next words(y_labels). Each batch(the number of batches is 100) has 5 words/time steps.

And then, one LSTM layer with 300 neurons/nodes is used. Different from char-wise model, all the five LSTM blocks/cells of this model generate outputs h0, h1, h2, h3, h4 whose shape is (100,5,300).

After that, there is a special Keras layer for use in recurrent neural networks called TimeDistributed. This function adds an independent layer for each time step in the recurrent model. For our model, we have 5 time steps in a model, a TimeDistributed layer operating on a Dense layer would produce 5 independent Dense layers, one for each time step. Each dense layer has 5201 units corresponding to the number of words in the vocabulary. And then softmax transform is applied to the outputs of dense layer which are called logits and generates probability distribution for each word. The word with the highest probability will be the predicted y which is the most suitable word to append to the input word sequence[5].

The rest steps are the same with char-wise model: calculate L and its gradient, backpropagation and update weights of each layer.

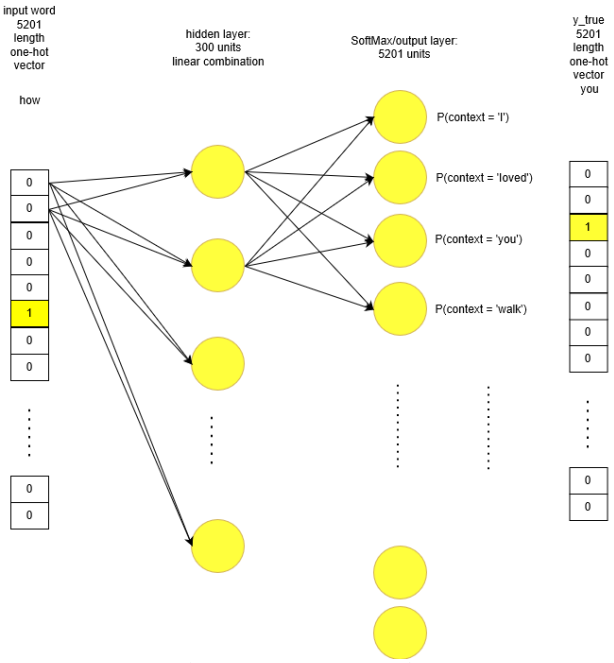


Figure 2 word2vec

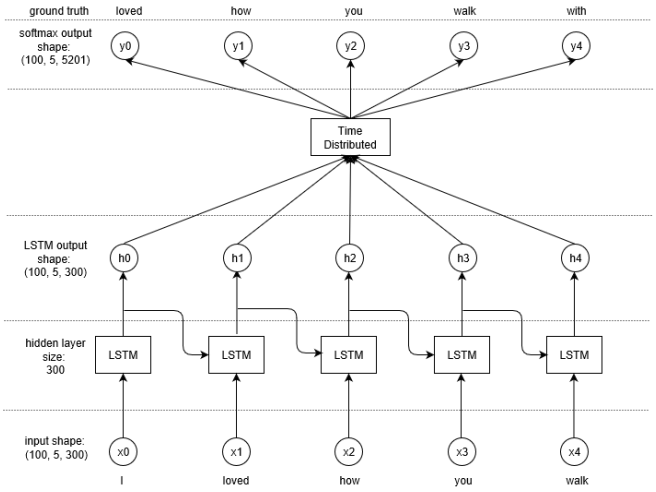


Figure 3 Word-wise model

c) Word-wise song-oriented model (bonus):

The second machine used for training was dedicated to experimenting with alternative parameters in a series of LSTM models. These parameters included experimentation with reduced normalization (capitalization and lemmatization omitted), character vs word-wise models, and varying the sequence length for both character and word based models (20 vs 40 characters, and 4 vs 8 vs 12 word sequences).

The intention for this alternative method was to produce ideas about how to improve the main model, and to optimize for subjective utility for using the software to create real songs. Since this was an entirely subjective measure, the results are only used for comparative and for entertainment purposes.

E. Result

1) Exploratory analysis results:

Table 1 lists the songs with the highest positive score and the songs with the highest compound score in some of the moods. It is shown that the song with the highest positive score in one mood does not necessarily have the highest compound score of that mood as well. For example, in the mood of angry, the song Anchor received the highest positive score, but the highest compound score of angry mood belongs to the song *It's So Easy*. It may be because the song Anchor has the highest proportion of words that are positive, but the sum of valence score of each word is lower than that of the song *It's So Easy*. Among all the moods, the song *She Loves You* has the highest positive score (0.73) (Figure 4) as well as the highest compound score (0.9994) overall. The word components of the song *She Loves You* is shown in Figure 5.

Table 1 Songs with the highest positive and compound score respectively in each mood (randomly display results of 6 out of 17 moods)

Mood	Highest Positive Score		Highest Compound Score	
	Score	Title	Score	Title
ambitious	0.62	Joy	0.9992	Joy
angry	0.642	Anchor	0.9971	It's So Easy
bittersweet	0.656	Ain't That Loving You	0.9992	Ain't That Loving You
fun	0.78	She Loves You	0.9994	She Loves You
happy	0.685	I Will Be True To You	0.9989	Love Isn't Easy(But It Sure Is Hard Enough)
sad	0.513	Tender Years	0.9962	Tender Years

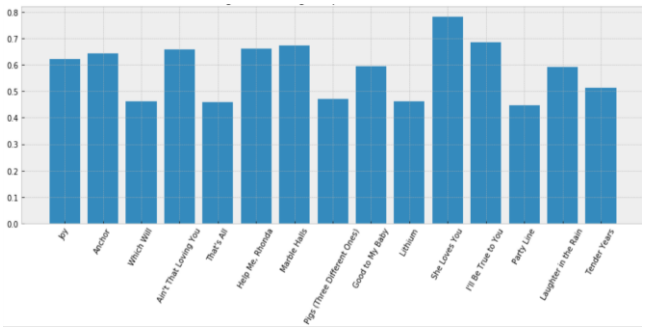


Figure 4 Songs with the highest positive score in each mood

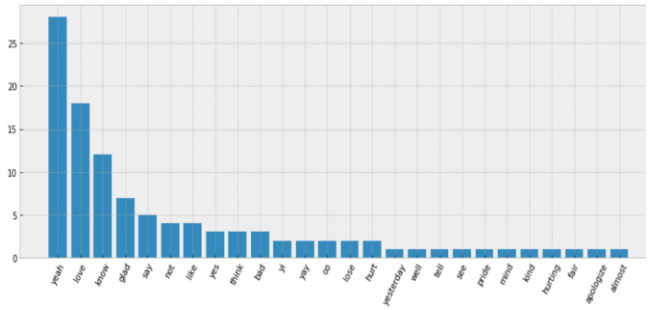


Figure 5 Frequency of words in the song *She Loves You*

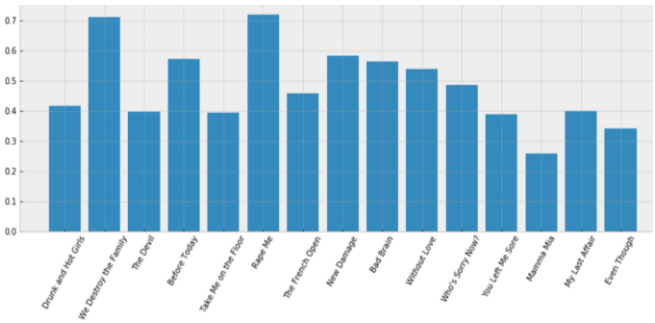


Figure 6 Songs with the highest negative score in each mood

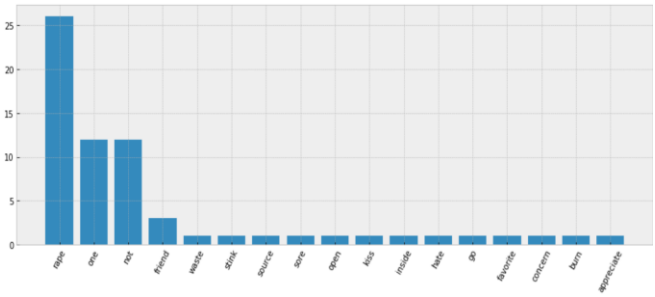


Figure 7 Frequency of words in the song *Rape Me*

In the same way, the songs with the highest negative score and the songs with the lowest compound score in each mood is compared. In one mood, a song may not achieve both the highest negative score and the lowest compound score, while the song called *Rape Me* obtained both the highest negative score (0.72) (Figure 6) and the lowest compound score (-0.999) of all songs in all moods. The word appears the most in the song is "rape", which appeared 26 times (Figure 7).

The sentiment scores of the set of lyrics of songs in the genre R&B (Rock and Ballad) is visualized in figure 8. The box plot shows the overall sentiment of genre R&B is positive since the compound scores lie on the positive side. (The mean compound score is 0.6914.) Though neutral words takes the greatest proportion, there are more positive words than negative words.

Figure 9 visualizes the sentiment scores of the songs of an artist named Billie Holiday and see how it changes throughout her albums. Based on the results, in terms of sentiment scores, it seems that most of her songs are either extremely positive, or extremely negative, as most compound scores are either close to 1 or -1. When compound score is negative, the negative score is greater than positive score, and vice versa, which makes sense.

The words with the highest frequencies used in each genre are also explored. Figure 9 and 10 show the example of looking for the 50 most frequently used words in the genre "Rap". In Figure 9, the word "brother" ranks at the 11th position, and the word "yo" ranks at the 13th position, which identify the word choice of rap lyrics. Figure 10 visualizes the word frequencies by the text size and color, that gives a stronger visual effect of the result to the audience.

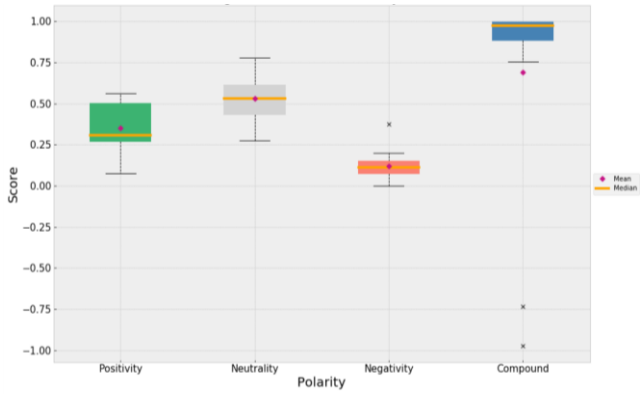


Figure 8 Sentiment scores for the lyrics set in genre R&B

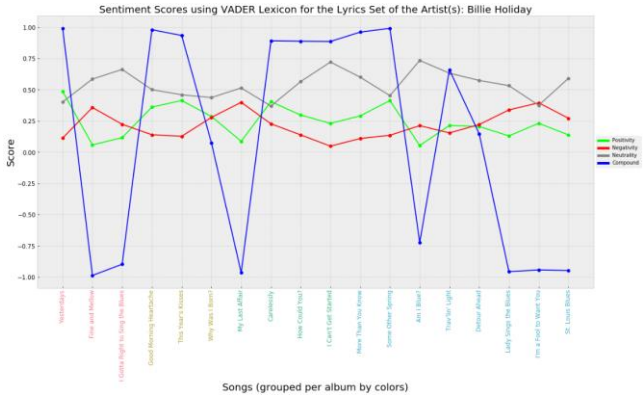


Figure 9 Sentiment scores for the lyrics set of the artist: Billie Holiday

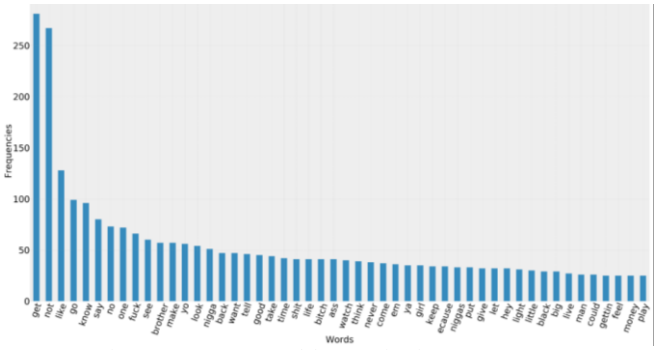


Figure 10 Top 50 word frequencies for genre: Rap

Figure 11 Top 50 word frequencies for genre: Rap (WordCloud)

2) Predictive analysis results:

At the beginning, the char-wise model generates Figure 12 which shows that exploding gradient problem appears causing the training process unstable.

Then, gradient clipping technique was used to keep gradient norm 1. Figure 13 shows that exploding gradient problem is resolved, but overfitting happens.

We simplify our model, utilize dropout technique and design a learning rate decade schedule to overcome overfitting problem. The final result of the char-wise model is shown in figure 14., and the result of word-wise model is shown in Figure 15.

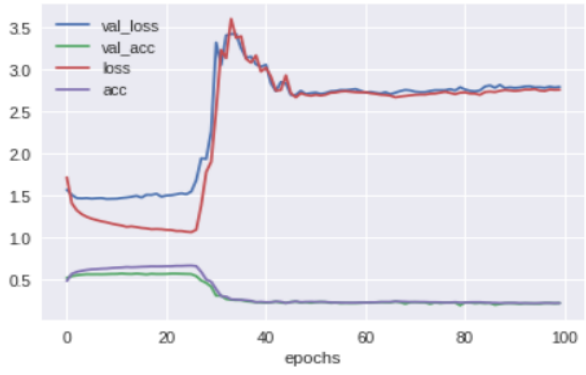


Figure 12 plot of char-wise model 1

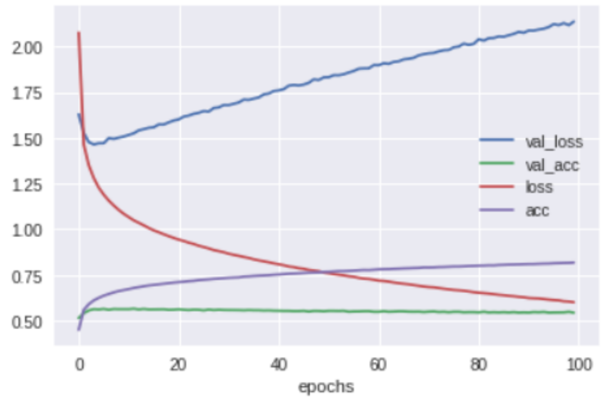


Figure 13 plot of char-wise model 2

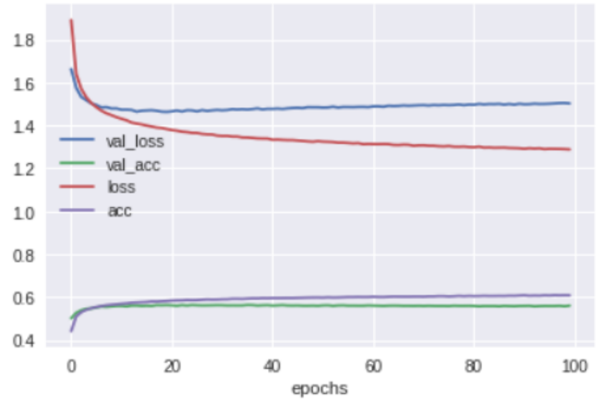


Figure 14 plot of char-wise model 3

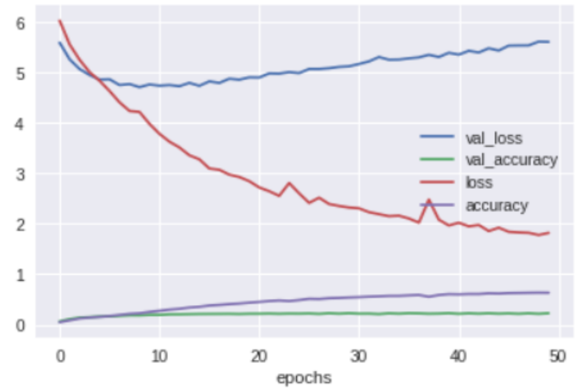


Figure 15 plot of word-wise model

Generating new lyrics with sample seed: "i loved how you walk with your hands in " for char-wise model.

Epoch 5:

i loved how you walk with your hands in the star the star i be a some be a star the star a star i be the star and a some the star when the star the star the stor i be the star i have be a back the thing i be a plast the star and i want to star the sear the star the thing i be go to be every she be the line when i be so the star the star a little the star i be go to be a baby be a some

Epoch 35:

i loved how you walk with your hands in the street to stay so star the baby the sing the tring can not say in the street and i be go to be the star the star i be a little and the star the sing the star the star of the star the say i be so star the star the back the time the light be and she be the star i will be the star the thing the and the star i be so time the world to the time the thing i be so star

Generating new lyrics with the same sample seed: "i loved how you walk with your hands in " for word-wise model.

Epoch 5:

I loved how you walk with your hands in the sky i be go to be a little i be go to be a little i be go to be a little i be go to be a little i be go to be a little i be go to be a little i be go to be a little i be go to be a little i be go to be a little i be go to be a little i be go to be a little i be go to be a little i be go to be a little

Epoch 40:

I loved how you walk with your hands in the crowd on the table let the good time roll find season at the wheel and the bright star of my heart be a laugh and a joke and a drink and a smoke and then i pass it out i will go to dance now i will never go to get it sized i will not get a lot of thing i want to talk about me i do i do i do i do i do i do i do i do i do i do i do i do i do i do i do i do

Word-wise song-oriented model:

As mentioned above, an additional LSTM model was trained in parallel on a different machine for exploratory purposes to focus on song-relevant lyrics. In this process, the loss function was not prioritized as highly.

For comparison purposes, the loss function was calculated for the first 20 epochs of one particular training run using 4-word sequences for training. It was less performant than the more complex, main model.

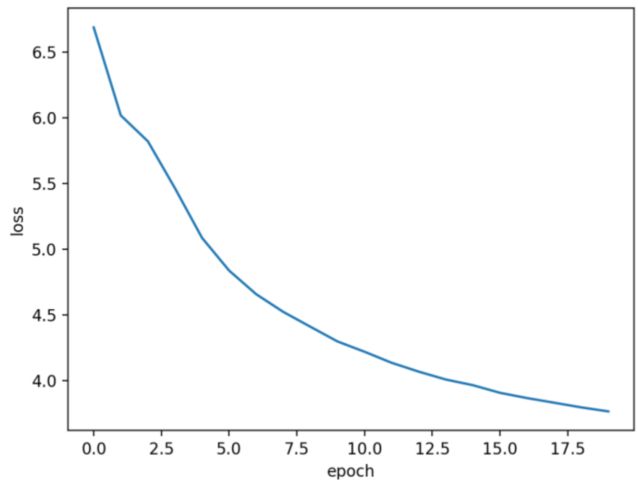


Figure 16 loss for the exploratory word-wise model with 4 word sequences

Clearly, the main model outperforms the 4-word sequence exploratory model. Yet another model was trained using 12 word sequences after this, and this model had substantially better performance with respect to the loss function, the lyrics were very difficult to put to music. When preparing the songs for the presentation, the 4-word model was used despite the relatively bad loss scores. It was subjectively far easier to make into usable songs.

Lyrics samples for experimental alternative model

(usable sequences for songs underlined)

“(...) Well, I just a little bit of time I need you to feel alright And the still love her mind I love I just the She with all my You can do it if you don't know why It's hard to will be a will you ever said oh So what love should if They won't come the back with you goes Don't the one more What a to get to love you feel this And I don't know the And I can't see me I need you to feel When I know you're gonna say”

"(...) the don't on the want to be your way I know you should be glad She should be think you're on Oh, I'm so my heart heart of the night is a I call my baby And she know that she was can see when I see the one I said I didn't know what to tell you How I miss that the people when you he turn around the before But you want to song) Your heart is Why you I can see you made you was gonna be good as you Why do I think you're too"

F. Discussion of the results

The minimal validation accuracy of char-wise model is 0.563 which is reached at the epoch 35. The minimal validation accuracy of word-wise model is 0.227 which is reached at the epoch 40.

Comparing the minimal validation accuracies of char-wise model and word-wise model, we can see that seemingly char-wise model performs better than word-wise model. But we should realize that char-wise model generates a character once while word-wise model generates a word once. If a word's length is 3, then the generation accuracy for char-wise model is 0.563*0.563*0.563=0.178 which is less than 0.227. And the lengths of most words are equal to or longer than 3. Therefore, generally speaking, word-wise model performs better than char-wise model.

For the generated lyrics by char-wise model: at the epoch 5, the generated lyrics still contain some unreal words; at epoch 35, the generated lyrics become better but are still not very good grammatically.

For the generated lyrics by word-wise model: at the epoch 5, there are a lot of repetition; at the epoch 40, though at the end of the generated lyrics there are some repetition, but generally the lyrics make more grammatical and semantic sense compared with epoch 5 and char-wise model.

Some of the lyrics generated were eventually put into music during an informal songwriting session. It was considered important to validate the work in this way due to the nature of the project's goal. In this case, the experimental song-oriented word-wise models were used for this. The character-wise models were too concerned with trying to spell words correctly to produce as interesting word sequences as the word-wise model. Although the main model outperformed this simpler model substantially in general, usable lyrics were generated during this session. A volunteer was found to sing these lyrics and these were put to music, producing the songs 'rap star Wuzza Wuzza' and 'Dangerous Woman,' among several others. It was found that models using shorter word sequences for training tended to be easier to put into a song in practice and this was surprising. We speculate that the shorter 4-word sequences simply fit into beats more easily, and that the more impressive longer strings tended to wander in somewhat-coherent, but distracting ways. This is however extremely subjective and is included mostly for completion sake.

The songs "Dangerous Woman" and "Rap God Wuzza Wuzza" were played during the project presentation as part of the discussion about system validation.

IV. Related Work

A. GPT-2 by OpenAI:

OpenAI is a non-profit institute attempting to ensure that artificial intelligence develops in a beneficial and ethical manner. In February of 2019, OpenAI completed work on a new text generation model called GPT-2. This model utilized 1.5 billion parameters and produces text is alleged to be so realistic that they opted to not publicly release the model. Instead, a lesser version of GPT-2 was publicly released that utilized only 117 million parameters.

In hopes of having a system to compare to, some popular songs were input into this pre-trained model in order to see if it could be used for such comparison purposes. It was deemed impractical to attempt to train a 117 million parameter model using a single GPU, so the pre-trained model was used. Although the resulting text is surprisingly realistic, it had tenuous links to the input text and was, as expected, not a song. The sample is below used a snippet of Marilyn Manson's 'The Beautiful People' as input and is included for contrast, if not for comparison:

Model prompt >>> And I don't want you and I don't need you
Don't bother to resist, or I'll beat you
It's not your fault that you're always wrong
The weak ones are there to justify the strong

The beautiful people, the beautiful people
It's all relative to the size of your steeple
You can't see the forest for the trees
And you can't smell your own s on your knees***

==========
SAMPLE 1
==========

, I don't want to help you," he says. "There's no point to you or to your little sister. There's no point to you and there's no point to you and let me see you."

"What do you want?" one of the women asks.

"You want to get your job done. You don't want to live under the constant scrutiny of the president; you don't want to feel the pressure to be accepted; don't want to think and live in America. I'm a mom. I'm just going to give you what you need, right now. I don't want to know you want to have your daughter's life back. You don't want people to tell you what you're supposed to do. So go."

The mother says goodbye to her son, then steps back into the house and runs to her father's car. As she gets back toward the car, the younger woman looks over her shoulder where she saw the white Hyundai. She took it in. She says, "It just looked so nice."

B. Difference:

Compared with our LSTM models, GPT-2 are much more complicated and training on more parameters. GPT-2 also takes punctuations and upper case words into consideration and generate more realistic lyrics.

V. Conclusion and Future Work

This paper proposed two LSTM models(character-wise model and word-wise model) to generate lyrics and a comparison study is done. The results show that word-wise model performs better than character-wise model and both can generate lyrics which make grammatical sense after training for a certain amount of epochs. Some of the lyrics generated were eventually put into music during an informal songwriting session which is used to validate the work in this way due to the nature of the project's goal, and these songs combining generated lyrics and audio are very fancy and impressive.

Future work:

- Webscrap and use larger dataset for different genres and moods to train LSTM models and improve their performance;
- Enhance preprocessing work to filter text noise wisely and keep more useful text information to improve generated text quality;
- Finetune parameters of these two models to improve lyrics generation;
- Explore, implement and compare other RNN algorithms(e.g. GRU) to find more suitable model(s).

References

[1] "A Practitioner's Guide to Natural Language Processing (Part I) — Processing & Understanding Text", *Towards Data Science*, 2019. [Online]. Available: https://towardsdatascience.com/a-practitioners-guide-to-natural-language-processing-part-i-processing-understanding-text-9f4abfd13e72 . [Accessed: 09- Apr- 2019].

[2] "dipanjanS/practical-machine-learning-with-python", *GitHub*, 2019. [Online]. Available: https://github.com/dipanjanS/practical-machine-learning-with-python/tree/master/bonus%20content/nlp%20proven%20approach . [Accessed: 09- Apr- 2019].

[3]*Comp.social.gatech.edu*, 2019. [Online]. Available: http://comp.social.gatech.edu/papers/icwsm14.vader.hutto.pdf . [Accessed: 09- Apr- 2019].

[4]"Simplifying Sentiment Analysis using VADER in Python (on Social Media Text)", *Medium*, 2019. [Online]. Available: https://medium.com/analytics-vidhya/simplifying-social-media-sentiment-analysis-using-vader-in-python-f9e6ec6fc52f. [Accessed: 10- Apr- 2019].

[5] "Keras LSTM tutorial - How to easily build a powerful deep learning language model - Adventures in Machine Learning", *Adventures in Machine Learning*, 2019. [Online]. Available: https://adventuresinmachinelearning.com/keras-lstm-tutorial . [Accessed: 09- Apr- 2019].

[6]"Word2Vec word embedding tutorial in Python and TensorFlow - Adventures in Machine Learning", *Adventures in Machine Learning*, 2019. [Online]. Available: https://adventuresinmachinelearning.com/word2vec-tutorial-tensorflow . [Accessed: 10- Apr- 2019].

[7] "Using AI to generate lyrics", *Medium*, 2019. [Online]. Available: https://medium.com/@ivanliljeqvist/using-ai-to-generate-lyrics-5aba7950903 . [Accessed: 10- Apr- 2019].

[8] "ivan-liljeqvist/ailyrics", *GitHub*, 2019. [Online]. Available: https://github.com/ivan-liljeqvist/ailyrics. [Accessed: 10- Apr- 2019].

[9]"Coursera | Online Courses From Top Universities. Join for Free", *Coursera*, 2019. [Online]. Available: https://www.coursera.org/learn/intro-to-deep-learning/home/week/5. [Accessed: 10- Apr- 2019].

[10]"openai/gpt-2", *GitHub*, 2019. [Online]. Available: https://github.com/openai/gpt-2 . [Accessed: 09- Apr- 2019].

Contributions

(alphabetical by last name)

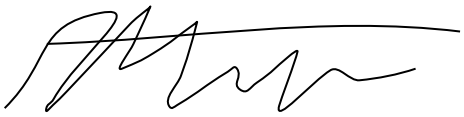
Jenard Cabia

- Applied sentiment analysis to the lyrics set of songs grouped by genre and artist
- Analyzed and visualized the results of song lyrics grouped by artist and genre as part of the Exploratory Analysis section in Phase 1 (Distribution of sentiment scores, Top 50 Word Frequencies)
- Completed the Introductory section of the report, listing primary experiment objectives, background, and solution

Shuyi Jin

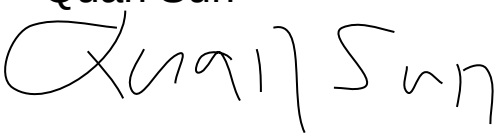
- Applied sentiment analysis to the lyrics grouped by mood and give sentiment labels to the lyrics
- Analyzed and visualized the most positive and the most negative songs in each mood and overall (Figure 4-7, Table 1)
- Completed data exploration, sentiment analysis technique and exploratory analysis result of the report and presentation slides

Nathan Schuetz



- Web scraping / data collection and munging
- Setting up tensorflow-gpu with keras
- Training alternative LSTM models with various parameters
- Producing song recordings, including music and singing, with help of a volunteer
- Various sections of the report
- Attempting to get the presentation under time budget
- Exploring other existing work (GPT-2)

Quan Sun



- Design LSTM char-wise model and word-wise model and code related python scripts
- Use Google Colab to design experiments on and train LSTM models, and generate new lyrics
- Code data preprocessing part scripts
- Make presentation slides and report contents about data preprocessing and LSTM models parts