ATL: Audio to Lyrics
CS 7643 / CS 4803 Deep Learning (Both Sections)

Jianni Adkisson
jadkisson6@gatech.edu
CS 7643

Marc-Henri Bleu-Laine
mhbl3@gatech.edu
CS 7643

James Thomas Jr.
jthomas8@gatech.edu
CS 4803

Abstract

Our work focuses on generating lyrics using deep learning models trained on audio and text and simply given audio and a starting word as input. Existing Research has been done to generate musical tracks for specific genres and put existing vocal tracks on top changing, as in OpenAI Jukebox. We explore the field of MIR and the math behind libaries like Librosa for working with audio in python, as well as combining text and audio data into one deep learning model. Using an audio captioning style model using RNN with LSTMs we generate lyrics to be classified by another neural network using RNN with LSTM as well.

1. Introduction

Generating new music takes creativity ranging from the instrumentals to the vocals to the lyrics. Lyrics can a tell a story, provoke an emotion, and determine the mood of the song. Machine learning focuses on identifying a pattern from its training data to build a model that categorizes or generates data. Deep Learning has become increasingly used in solving complex NLP Problems. We wonder if we could use DL techniques to replicate this creative process of generating lyrics from raw audio.

The goal of this project is:

1. To develop a deep learning model to perform audio to text generation
2. To combine raw audio and lyric data to generate lyrics for Country and Hip-Hop songs
3. To classify newly generated lyrics as belonging to Country or Hip-Hop genre

1.1. Background

Traditional approaches involved using non deep learning techniques to classify a song's genre using its lyrics. In [4], this work involved classifiers such as decision trees, random forest, k-nearest-neighbor and Naive Bayes algorithm to classify a song's genre. They observed 5 genres: rock, rap, country, jazz and pop. Random forest had the best accuracy with a 52%.

In recent years, deep learning techniques have become popular in solving Natural Language Processing problems. Our work will focus on Deep Learning techniques such as RNNs for genre classification. In the work[14], they applied recurrent neural network models to classify the music genre using song lyrics. They adapted a hierarchical attention network and compared this model with non neural models.

Jukebox AI[6] use generative models such as VQ VAE for music generation. The model generates music with singing in the raw audio domain. They condition their model on the lyrics corresponding to each segment which allows the model to produce singing simultaneously with the music. Our work will do something similar. Our work will use both the audio and lyrics to train a model to generate new lyrics.

Additional deep learning application to audio include audio tagging and audio captioning, sound event detection, and more. In particular, the work proposed by Kong et al. [9] is of interest for this work as its contribution allowed for the creation of a data set of Pre-trained Audio Neural Networks (PANNs) for audio patterns recognition that could be transferred to other audio related tasks. Furthermore, the authors investigated the performances and computational complexities of different algorithms for audio tagging tasks for different type of audio inputs such as audio wave forms and log-mel spectrogram. We will cover the Mel Spectrogram in sufficient detail shortly.

Regarding audio captioning, Eren et al [17] extracts log mel spectrograms from raw audio data. The extracted spectrograms are embedded using a VGGish model [8] and the embeddings are encoded with Bi-directional Gated Recurrent Units (BiGRU). Text caption are also embed using Word2Vec [10]. The two data type embeddings are then combined and pass through a GRU decoder for the next word prediction.

1.1.1 Music Information Retrieval

Music information retrieval (MIR) is the science of effective extracting, searching, and organization of music information. There are many techniques from a broad range of fields that we can try to utilize in order to help turn the music into something more feature rich for our models to work with. Some fields that intersect with MIR are musicology, digital signal processing, machine learning, information retrieval and library science. The International Symposium on Music Information Retrieval (ISMIR) has been going since 2000. For a better understanding and more modern approaches to MIR check out their website [1].

When approaching music and audio analysis in python we are typically going to be working with Librosa primarily [3]. Librosa is a library built on the idea of standardizing core functions in music information retrieval and it was very successful at that. Today we see that Librosa is used in many if not most MIR papers. The Mel Spectrogram function using the is of primary focus, however for future work the library provides other methods for input/output and digital signal processing.

1.1.2 Fourier Transform and Spectrograms

Briefly, Integral transforms were invented by Euler and we can see it being used as early as 1763 in his work, however of more interest to the reader might be Laplace's transform. Laplace wrote about his transform in his paper Essai philosophique sur les probabilités from 1814. His studies were primarily concerned with generating functions. In particular moment generating functions, those used to determine the moments (expectation, variance, ...) of a distribution. Integral transforms take an equation in one domain and change it into another domain. This can work in reverse as well. In particular the domain that we reach using distributions end up telling us the hyper parameters. However the inverse Laplace transform is rough. Not everyone understands contour integration, so in many differential equations classes we will simply be provided with a table for the inverse transformations.

On the other hand, Fourier was more interested in keeping his house warm. He was concerned with figuring out the distribution of heat around a ring and determined it had something to do with the sinusoidal frequencies within it. Fourier transforms are much easier to deal with as well. Not only that but not all distributions will have moment generation functions from the Laplace transformation. However, all distributions will have a characteristic function that is generated from the Fourier transform. There is a rich history behind signal processing with many ideas coming from these probabilists. Returning to our current topic of MIR and signal processing, the Fourier transform provides a tangible way of analyzing, meaning to say that the Fourier transform allows us to capture the steady state characteristics of the audio and disregard transient features in the complex domain that the Laplace transform would offer. In a sense, we are performing dimensionality reduction by choosing the Fourier transform instead of the Laplace transform for analysis. Laplace transforms are better suited for systems analysis, and with Fourier transforms are best suited for obtaining the frequencies in a signal.

The heavy lifting of the Mel Spectrogram is done by the Fourier transform, in particular the short-time Fourier transform. The Fourier transform is an integral transform of the form:

$$\int_{-\infty}^{\infty} f(t) e^{-2\pi i u t}\, dt \tag{1}$$

where f(t) is our song - however music can have multiple tracks or channels. For simplicity will keep the discussion to a single channel for now. What we have seen so far was for a continuous signal however, discretizing the Fourier transform (DFT) we obtain:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i 2\pi k n / N} \tag{2}$$

Further more our short-time Fourier transform will also be of the discrete form:

$$X(m, w) = \sum_{-\infty}^{\infty} x[n] w[n-m] e^{-jwn} \tag{3}$$

where $w(n)$ is a window function that is only nonzero for a short period of time (there are many different window functions to choose from each with the upsides and downsides, but for simplicity's sake we can view a window function as the difference between two Heaviside functions). The spectrogram of a window is the discrete Fourier transform (DFT) of a signal in that window, in our case we would end up with a matrix, given multiple tracks and multiple DFT for the same window. So converting our audio into a spectrogram means generating a heat map based on the intensity of the frequency, with the sample time running along the x-axis as it would in a wave file but the frequency ranging along the y-axis.

1.1.3 The Mel Scale

The Mel Scale is a perceptual scale of frequencies[15]. Listeners were asked to judge the perceived distances and the scale was created. An argument can be made that we are losing information for being processed by our model that perhaps a machine would work better with. However, since the music is generated by humans, the information could be potentially unintended or over complicating our model.

There is also no common formula for the Mel Scale. The formula for converting Hertz into Mels is:

$$m = 2595log_10(1 + f/700) \quad (4)$$

We simply feed our frequencies from our spectrogram into this equation and we end up with the Mel Spectrogram which we then use as inputs into our models.

1.2. Motivation

Musicians, like writers, can face writer blocks and may lack inspiration at some point in their careers. This can be stressful the artist and lead to anxiety and self-doubt [16]. There exists tips on how to deal with writer block such as creating an environment for creativity, writing something else than a song, exercising, etc. What we propose is a novel way for musicians to get started or continue their work, and potentially unlock their block by injecting them AI generated creative juice. In particular, our model is ideal for artists that have instrumentals (i.e. an audio clip) but are having a hard time with a song that they are writing. We allow them to get lyrics directly inspired by the music that they are trying to sing on.

1.3. Data set

As part of an effort for this project, a custom data set was built. The data set creation is a four steps process that includes:

1. Top artist search for music genre of interest
2. Spotify playlists creation
3. Tracks download
4. Lyrics download

This process allowed for the creation of balanced training data set with well known English artists and fairly recent tracks. Additionally, balanced validation and testing sets were also created. The data set creation first start by obtaining a list of the top 10 artists for both rap and country music. For this project, the top artist names were retrieved from the billboard 2019 year-end top artists [1]. The second step of the data creation process was to create Spotify playlists, for both music genre. The Spotipy [2] library offers the ability to connect to the Spotify's Web API through Python and gather music data. Spotipy is useful to automate the collection of song titles for artists of interest and the creation of Spotify playlists. Thirdly after collecting 10 song titles per artist, we leveraged the Youtube_search [3] and the Youtube_dl [4] libraries to search and download tracks, from Youtube, corresponding to the ones in the created Spotify playlists. Finally, the Python API for the Genius website was used to find lyrics for the songs that were successfully downloaded resulting in a custom data set including tracks of multiple artists along with the lyrics for each track.This data was used to train the lyrics generating model.

Additionally, we used a MetroLyrics dataset [5] from Kaggle to train the genre classifier, which included the song name, lyrics, artist and genre. We would only use the Hip-Hop and Country lyrics.

1.4. Tools

The primary tools used for this project was Python, and all the different libraries:

- Pytorch
- Keras
- Scikit-Learn
- Librosa
- Youtube_dl
- Youtube_search
- Spotipy

2. Approach

Our approach was inspired by audio and image captioning (seen in class) which are the process of generating textual description from an audio clip and an image, respectively. We believed that we could generate lyrics in a similar manner using 10 seconds audio clips of particular music genre and their corresponding lyrics. We therefore developed and trained deep learning models that used log mel spectrograms [11] embeddings to predict the next word from the lyrics of the song corresponding to the input audio. Given how successful image captioning use cases are, the application of similar approaches to audio data did not seem like a far stretch and therefore was expected to be successful as well. Moreover during training, we combine the embeddings of text and audio inputs using an addition operation. While this combination, and the generation of the embeddings is not novel, there exists little to no work on the use case that we investigated (i.e. lyrics generation based on audio input). After generating new lyrics, we pass this information to the genre classifier that will output the genre probability. This step was necessary to allow us to get an idea of "how country/hip-hop" is the lyric generated. The genre classifier enables a some semblance of quality check.

The major issues encountered in this approach were related to scalability, and lack of experience processing audio data. In fact, the very first model developed for the text generation tasks was able to train on a few examples. However,

[1] https://www.billboard.com/charts/2019/year-end
[2] https://github.com/plamere/spotipy
[3] https://pypi.org/project/youtube-search-python/
[4] https://pypi.org/project/youtube_dl/
[5] https://www.kaggle.com/danofer/music-lyrics-clean-export

as we extended training to more and more examples, we ran into RAM issues. Also, training the models took too long, preventing us from being able to quickly iterate. The memory overhead is due to the log mel spectrograms that were extracted. Pre-trained models available thanks to [9], were also attemped but the models never finished training before the he runtime failed.

2.1. Data Preparation

2.1.1 Text Generation

For the text generating model pre-processing was required for both the text (lyrics) and audio clips. A standard text processing approach was applied to the lyrics. The text was converted to lowercase, all the punctuation were removed, words of one character, and words with number were removed. The cleaned lyrics were tokenized resulting in a vocabulary size of 5690. Finally a mapping from word to index was created. Each sentence was then converted to a one-hot encoded vector with length equal to the longest sentence in the availeable data.

The audio inputs were re-sampled to 16 KHz similar to [17] and [9] using the Librosa library. CD audio quality is usually is sample at 44 KHz. The sampling rate used for our work is due to the fact that working with music data is extremely computationally demanding [6]. A full list of the parameters used to extract the log mel spectragram is presented in table 1

Table 1. Parameters Used for Log Mel Spectrogram Extraction

Parameter	Value
Sampling Rate	16000 Hz
Audio Clip Duration	10 Secs
number of FFT components	512
Hop Length	160
Mel Filters	64

Finally, the text and audio are combined such that for a given audio clip the next word for each lyric sentence is predicted by the model. Table 2 shows an example of the input/output setup for this project.

2.1.2 Lyric Genre Classification

Metrolyrics provides the lyrics to various songs from different artists and genres. We used the MetroLyrics Dataset from the Kaggle's dataset library. Since the Kaggle dataset wasn't currently available, we used the dataset from [5]. This dataset contained the post-processed and cleaned song lyrics originally obtained from the MetroLyrics Kaggle dataset.The dataset was filtered to only include Country and Hip-Hop songs.

Table 2. Parameters Used for Log Mel Spectrogram Extraction

Input Text	Input Audio File Name	Target Text
startseq	Drake-Toosie Slide.mp3	black
startseq black	Drake-Toosie Slide.mp3	leather
startseq black leather	Drake-Toosie Slide.mp3	glove
...	...	...
startseq basically im sayin either way we bout to	Drake-Toosie Slide.mp3	slide
startseq basically im sayin either way we bout to slide	Drake-Toosie Slide.mp3	endseq

2.2. Text Generation

The model selection and the choice of hyperparameters used were heavily influenced by the work proposed in [9], [17] and [13]. However, in order to allow for faster training of the model, a simpler architecture was implemented. The text inputs are embedding, using an embedding layer, then encoded using stacked LSTMs are seen on fig. 1. The output of the encoding is then flattened. The input is encoded using a series of 4 convolutional blocks followed by a dense layer. Combining the encoded text and audio using addition allows for model to learn to associate the lyrics with particular audio sounds. The loss function to train the model is the cross entropy function implemented in Pytorch given

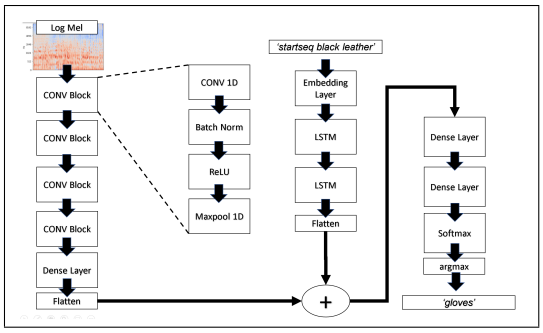

Figure 1. Encoder for Lyrics genrator

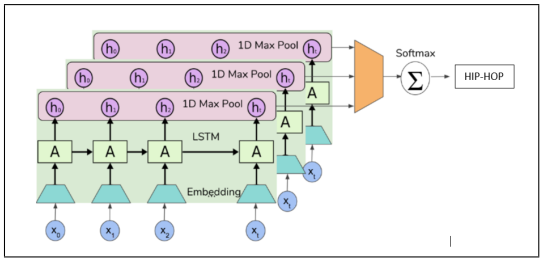

Figure 2. Architecture for Vanilla LSTM RNN

by:

$$loss(x, y) = -\log\left(\frac{\exp(x[class]}{\sum_j x[j]}\right) \quad (5)$$

All the convultuional blocks use a kernel size of 3 with a stride of 1, similar to the CNN10s model in [9]. The output channels for each block is steadily multiplied by 2 starting from 64 to 512. The text is embedded using a embedding vector of size 128. The latent dimension used to combine the encoded audio and text inputs is 256.

The combined encodings are then decoded by two dense layers layers followed by a softmax that predicts the probability of the next word.

2.3. Lyric Genre Classification

Since we are performing binary classification, we used a basic vanilla LSTM to implement the genre classifier. We followed a similar approach to develop the LSTM model as [5]. They wanted to classify the lyrics with 11 genres and they found that their baseline LSTM model had similar performance to the other models they implemented. We found that since Hip-Hop and Country were very different that a simple architecture would be enough. The LSTM architecture as summarized in 2:

$$Lyrics \rightarrow Embedding \rightarrow LSTM \rightarrow MaxPool$$

$$\rightarrow SoftMax \rightarrow Probability$$

We chose a softmax activation layer so the output of the network would be probabilities. Given some lyrics, the classifier will classify the probability the lyrics are Country or Hip-Hop and output the predicted genre label.

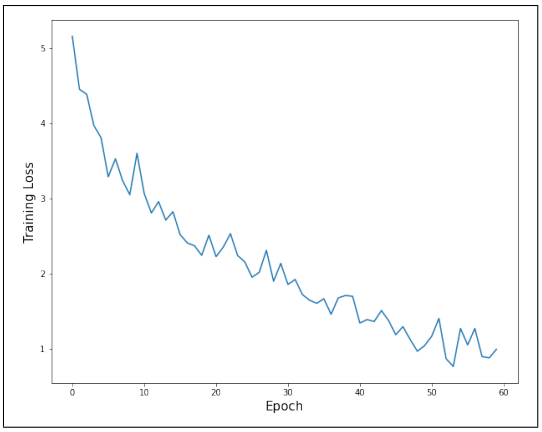

Figure 3. Training Loss for Text Generator

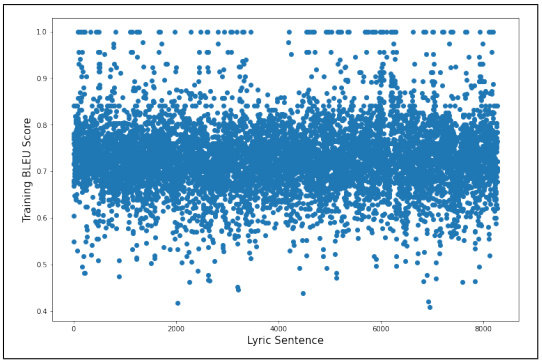

Figure 4. Training BLUE Score for Text Generator

3. Experiments and Results

(10 points) How did you measure success? What experiments were used? What were the results, both quantitative and qualitative? Did you succeed? Did you fail? Why?

3.1. Text Generation

After training the model for 60 epochs, (2 hours long training) using the ADAM optimizer. The model performance was evaluated using the loss at every epoch as seen on fig3. As seen on the figure, the loss is behaving as expected and the curve suggest that further training the could could yield better results. The BLUE score (fig. **??** was also computed to see how we the model generates lyrics based on audio and the starting word for each sentence in the lyrics.

Overall the model shows that it is able to learn from the data, this can be seen also in generated lyrics by the model. However When assessing the quality of the generated lyrics for the validation and testing sets, we realized

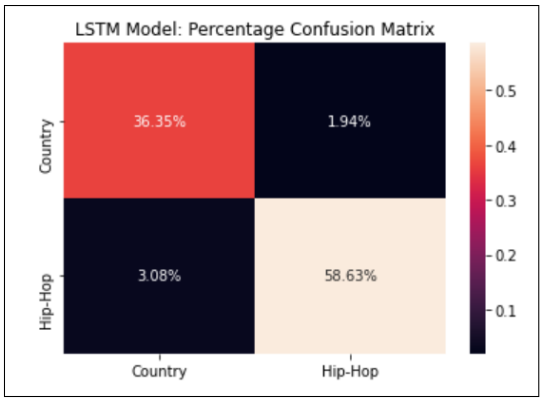

Figure 5. Confusion matrix for genre classification LSTM model

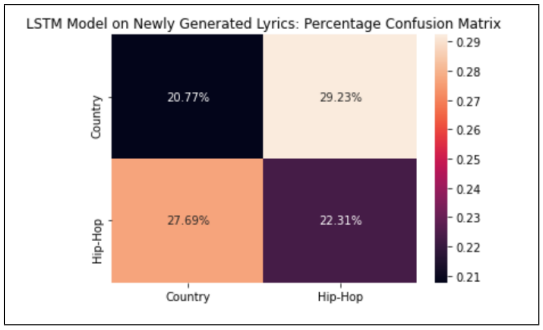

Figure 6. Confusion matrix for genre classification LSTM model on newly generated lyrics

that the model learned to predict "outro" when given the startseq token. The discrepancy between the lyrics generated based on training audio and the lyrics generated based on validation/testing audio suggest that the model overfitted. The model generated lyrics on new audio are better when using starting words or sentence other than startseq though the semantic of the lyric is questionable at time.

3.2. Lyric Genre Classification

3.2.1 Real Data

The LSTM model achieved a 95% accuracy on the test dataset from the MetroLyrics.

The model is able to classify the genres well because of how distinct they are. The percentage confusion matrix in 5 shows the percentage of the data represented in each quadrant. The LSTM model successfully labels 59% of the data as Hip-Hop and 36% of the data as a Country. It misclassifies 5% of the data.

3.2.2 Synthetic

The LSTM model however was not able to classify the genres as well with the newly generated lyric from our text generation model. The model typically confused the two genres when using the synthetic data as show in 6 . The LSTM model successfully labels 22% of the data as Hip-Hop and 21% of the data as a Country. It misclassifies 57% of the data. As you can see a lot of the lyrics were mismatched. Typically, the more explicit lyrics were labeled as rap even though the text generator labeled it as country. This suggest the text generator model can be improved to produce better lyrics for the genres.

4. Conclusion

In this work, we showed that AI generator lyrics based on audio inputs is possible and could help musicians facing writer blocks. After reviewing past work applying deep learning to audio and use our knowledge of image captioning, we created a model that was able to create somewhat acceptable lyrics. However, the model overfitted, possibly due to multiple reasons such as the audio data preprocessing, the data set size, and the usage of non-optimal parameters. This short-comings can be addressed in future work. Other models such as the pre-trained CNN14s [9] were used in an attempt to train, however the final trained models were never obtain to the run-time crashes.

5. Future Work

5.1. FFT alternative: Constant-Q Transform

An alternative to the fourier transform for spectrogram analysis that has become less computation expensive recently is Constant-Q Transform [12]. Constant-Q transform is very similar to the short-time Fourier Transform however instead of taking uniform windows at all frequencies, it takes larger windows for lower frequencies and smaller windows for higher frequencies. This varying buffer sizes are helpful to understand more dynamic changes at the higher frequencies and more accurately classify lower frequencies with larger windows. All this can actually be done in $0(N \log N)$ time, the same as FFT [2].

5.2. GAN

After considering the best transform for MIR, another concept to investigate would be trying to train our entire ecosystem with a GAN setup. GANs can work well for audio and visual data, but the problem lies more in the discrete nature of NLP. For generative models we have seen that VAE can handle this task well, but in general GANs tend to have more appealing results. Recently we have seen more breakthroughs in text generation using GANs, despite the models continuous nature and the I/O's discrete nature conflicting with each other [7]. The secret sauce would be converting the text to a continuous distribution using an autoencoder. This would be really excited to see work, but will likely take some time. With the text GAN we would ideally be able to incorporate its features into a GAN for

our system, with our classifier as part of our loss function and incorporating the audio as input as well.

References

[1] Ismir. 2

[2] Peter Balazs, Monika Dorfler, Nicki Holighaus, Florent Jaillet, and Gino Angelo Velasco. "theory, implementation and appplications of nonstationary gabor frames". 6, 7

[3] McFee Brian, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, , and Oriol Nieto. "librosa: Audio and music signal analysis in python.", 2015. In Proceedings of the 14th python in science conference, pp. 18-25. 2

[4] Anthony Canicatti. Song genre classification via lyric text mining. 2016. 1

[5] Brennan Connor, Paul Sayan, Yalamanchili Hitesh, and Yum Justin. Classifying song genres using raw lyric data with deep learning, 2018. 4, 5

[6] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music. 2020. 1, 4

[7] Md. Akmal Haidar and Mehdi Rezagholizadeh. Textkd-gan: Text generation using knowledgedistillation and generative adversarial networks, 2019. 6

[8] Shawn Hershey, Sourish Chaudhuri, Daniel P. W. Ellis, Jort F. Gemmeke, Aren Jansen, R. Channing Moore, Manoj Plakal, Devin Platt, Rif A. Saurous, Bryan Seybold, Malcolm Slaney, Ron J. Weiss, and Kevin W. Wilson. CNN architectures for large-scale audio classification. *CoRR*, abs/1609.09430, 2016. 1

[9] Qiuqiang Kong, Yin Cao, Turab Iqbal, Yuxuan Wang, Wenwu Wang, and Mark D. Plumbley. Panns: Large-scale pretrained audio neural networks for audio pattern recognition. *CoRR*, abs/1912.10211, 2019. 1, 4, 5, 6

[10] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546, 2013. 1

[11] Lawrence R. Rabiner and Ronald W. Schafer. *Theory and applications of digital speech processing*. Pearson Higher Education, 2011. 3

[12] Christian Schorkhuber and Anssi Klapur. "constant-q transform toolbox for music processing", 2010. 6

[13] Marc Tanti, Albert Gatt, and Kenneth P. Camilleri. Where to put the image in an image caption generator. *CoRR*, abs/1703.09137, 2017. 4

[14] Alexandros Tsaptsinos. Lyrics-based music genre classification using a hiearchical attention network. 2017. 1

[15] Umesh, Cohen, and Nelson. Fitting the mel scale, 1998. In Proc. International Conference on Acoustics, Speech, and Signal Processing (ICASSP), vol. 1, pp. 217-220. 2

[16] Matt Williams. How to beat writer's block – spotify for artists, Aug 2019. 3

[17] Ayşegül Özkaya Eren and Mustafa Sert. Audio captioning using gated recurrent units, 2020. 1, 4

6. Appendix

6.1. Experiment Details

6.1.1 Algorithms

As described in the main section under the approach, the models used Lyric Genre Classification and Text Generation was described in detail.

DFT that does the heavy lifting for the spectrogram has the time complexity of $O(NlogN)$. [2]

6.1.2 Source Code

Our source code is accessible at https://github.com/mhbl3/audioToLyrics.git

6.1.3 Dependencies

In python3 you will need the following up-to-date libraries:

- jupyter notebook
- pytorch
- keras
- librosa
- lyricsgenius
- spotipy
- youtube_dl
- youtube_search
- JayChen35/spotify-to-mp3-python.git
- pandas
- tensorflow
- sklearn
- seaborn
- matplotlib

6.1.4 Runtime

The average training time for the text generation model was 2 hours.

The average training time for the lyric genre classifier was about 20 minutes.

6.1.5 Parameters

Preprocessing Mel Spectrogram parameters (using librosa): Sample rate: 16000 Hz, Audio Clip Duration: 10s, number of FFT components: 512, Hop Length: 160, Mel Filters: 64

6.1.6 Validation Performance

To measure the performance of the LSTM model for lyric genre classification, a confusion matrix was created to display how accurate the model was at classifying the genre from the test and synthetic dataset. Then, a csv file was additionally observed that showed how the model performed on the synthetic lyrics and the probability of the predicted genre. This also validated the text generation model performance to observe if the lyrics outputted for the specified genre match the genre predicted from the genre classifier.

6.1.7 Datasets

For the text generation model, a custom data set was built. The data set creation included searching for top artist search for music genre of interest, creating a Spotify playlists, and downloading audio tracks and lyrics. More details of this process is highlighted in 1.3 Data Set.

There was no pre-processing done for the genre classifier as we had access to a dataset that completed the preprocessing mentioned in 1.3 Data Set. The dataset for the genre classifier included around 37,000 songs. Out of those songs about 14,000 were Country and 23,000 were Hip-Hop. The training and testing was split up into 30,000 training data and 7,000 testing data. The newly generated lyrics consisted of 65 lyrics for each genre.

Additionally, the output from the text generation model was used as input for the genre classifier. Place the resultant csv file from the text generation model output in the dataset folder and run the last cell block in the DL_LyricsToGenre.ipynb. This will classify the genre of this synthetic data. The dataset we used is currently in the dataset folder on github but one can replace this file when rerunning the code.

6.1.8 Infrastructure

Google Colab was used along with Google Cloud Platform Compute Credits.We also access data using Spotify's API through python.