A project report on

# LYRICS GENERATION USING NEURAL NETWORK

Submitted towards J component of the course

**Artificial Intelligence CSE3013**

Under the guidance of

Prof. Santhi K

**Submitted by-**



| | | |
|---|---|---|
| 1 | RITIKA SINGH | 19BCI0166 |
| 2 | SAURABH SINGH | 19BCI0184 |



VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

# MOTIVATION

- The purpose of this model is that unusual and creative arrangements of words in the generated lines can inspire the songwriter to create original lyrics at a later stage.

- The reason for using such generative models is mainly to augment the natural creative process when an artist gets inspired to write a song based on something they have read or heard.

- The motivation for this project was to create new and interesting lyrics for artists that people love or that no longer write song.

- We wish to create a generator program which is successfully and opportunely able to generate the desired lyrics of the song with ninety percent accuracy along with bringing melody to the music and building the theme of the song in order to touch the hearts of the public.

- An important aspect of writing song lyrics is that it is quite difficult to create a song whose lyrics enthral the public and that can generate a positive response in the majority of the crowd. These aspects make writing lyrics for a song hard quite a challenging task for a singer. Keeping this problem in mind, we have decided to create a program called – Lyrics Generator.

# OBJECTIVE OF THIS PROJECT

- Generating artist-like lyrics that are needed to prepare new songs.
- Making use of neural networks and corpus in order to train and generate the lyrics .
- Giving the user the facility to generate lyrics by providing only one word related to his/her topic .
- Generate a self-made rap using our model .

# ABSTRACT


- A lyricist has a very challenging job that tests his creativity on every level. He must go through different aspects before giving his final track the complete lyrics. These aspects include choosing words that ideally rhyme with regard to the title of the song and that can give more meaning and sense in order to add to the theme of the song. Another important aspect of writing song lyrics is that it is quite difficult to create a song whose lyrics enthral the public and that can generate a positive response in the majority of the crowd.

- These aspects make writing lyrics for a song hard quite a challenging task for a singer. Keeping this problem in mind, we have decided to create a program called - **Lyrics Generator**.

- The concept of **Natural Language Generation** techniques are used in this program. A part of the desired song is given as input to the program and the program then gives its output as a song containing words from the part.

- This generator program is also capable of giving us words that rhyme with our song. We wish to simplify the task of lyricists and help them get their much needed inspiration to create the best quality of music.

# LITERATURE SURVEY

| | Title of the Paper And year | Algorithms Used | Performance Measures | Dataset being used | Gap identified | Scope for future work |
|---|---|---|---|---|---|---|
| 1 | **Base Paper- Generation of lyrics using RNN(2020)** | A model using RNN and LSTM networks where the grammatical errors, Spelling mistakes and special symbols were greatly reduced. | Computing the probability of occurrence of each letter from our training corpus which means the number of times a letter occurs divided by the size of the given dataset and randomly sample letter using these probabilities. | The data set was collected from an online song lyrics database. | Large execution time and low accuracy. | Using more data to train our model, making it easier for the system to actually identify rhyming pairs. |
| 2 | Dope Learning: A Computational Approach to Rap Lyrics Generation. (2016) | Created a lyrics generation model using two techniques: Rank Support Vector Machine and Deep Neural Network | Next line prediction is the method used. After feature extraction, both bag-of-words and Latent semantic analysis is used to find similarity and then passed through a neural network to train. | 10980 songs from 104 different artists were used as the dataset. The dataset was passed through pre-processing techniques like removal of stop words, tokenization, etc | Since a lot of operations were done on the data, the execution time is very large and in accordance to that, the accuracy obtained is not enough. | It would be interesting to study automatic creation of story lines by analyzing existing rap songs and of novel lines by modifying existing lines or creating them from scratch. |
| 3 | GhostWriter : Using an LSTM for Automatic Rap Lyric Generation. (2015) | Created a lyrics generation model using LSTM. | The dataset is passed through an LSTM model which is used to generate lyrics. The results are tested for their similarity with original ones and their rhyming density. These values are 3compared w4ith a simple n-gram model to test the accuracy. | Songs from 14 artists from The Original Hip-Hop (Rap) Lyrics Archive were used as dataset. | The value obtained for similarity index and rhyme density comes out to be smaller than n-gram because of repetition of words. This happened because pre- | Plan to use more data to train our model, making it easier for our system to actually identify rhyming pairs and use them in new contexts. |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | processing techniques were not implemented on the data. | |
| 4 | Generating lyrics with variational autoencoder and multimodal artist embeddings. (2018) | Lyrics generation using CNN classifier and variational Autoencoder (VAE) along with LSTM. | CNN classifies artists on the basis of spectrogram images, then VAE is trained to construct lines from song lyrics from the artist embeddings obtained from CNN which uses LSTM for encoding and decoding. Finally the lyrics are decoded to generate the final lyrics | Dataset contains songs from 7 artists, one from each genre: Art Rock, Industrial, Classic Rock, Alternative, Hard Rock, Electronic, and Psychedelic Rock making a total of 34000 lines. | The accuracy is lower when compared to other models. Also, the execution time for the model is high because of Variational Autoencoding, LSTM and then finally decoding the lyrics. | Plan to evaluate other models for pre-training of artist embeddings, for example spectrogram autoencoders. Also explore other approaches to learn multi-modal representations, e.g. [14] and adversarial approaches. |
| 5 | Video-Based Emotion Recognition using CNN-RNN and C3D Hybrid Networks. (2016) | The main aim of the paper is to find the emotion related to a particular video. RNN takes appearance features transferred by implementing CNN on particular video and C3D models appearance and motion of video. | The system is a hybrid network which has 2 core parts: RNN-CNN and C3D. Audio classifiers are also added to the system so that the system can develop the ability to differentiate between sounds to complete specific tasks. LSTM is also introduced in the system to deal with the vanishing gradient problem and later C3D structure is used as a spatial-temporal model. As far as implementation6goes all the faces of vieo | AFEW 6.0 database is taken into consideration which contains 1750 video clips that were further divided into three parts: 774 for training, 383 for validation, and 593 for test. | With the use of improved models, the accuracy of the system can be improved. | In this paper only 2 models are combined C3D and CNN-RNN which produces effective results, but the combination of more than 2 models the accuracy can be improved. |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | frames are extracted and aligned using a similarity transform according to the facial key points and a CNN filter is applied to filter out the mistakes of faces. | | | |
| 6 | Deep Learning based Poetry Generation Given Visual Input. (2018) | The main idea of the framework is to generate poetry from the given visual input. The system will consist CNN for image classification and a LSTM neural network trained on a song lyrics data set. | The basic idea of the architecture is, once the system gets the input it will go through inception for classification. Then inception will produce 5 top scoring class and if these 5 classes are below the threshold class then the input will be discarded otherwise the image is carried to the next stage where the related words are discovered related to the image using the concept called Concept Net Before displaying the poetry 2-3 steps are carried out to make the process optimal and efficient. | The data set was collected from www.mldb.org, an online song lyrics database. To connect to site the authors had to write a python script and sort through the HTML files of the site and find the song text, artist and album. | One thing that was clear from the output was that the system was not able to generate meaningful poetry i.e. it lacked sense. | To enhance the predictions sequence to sequence model can be used instead of LSTM. To get the more accurate result training of LTSM can be done on poetry rather than song lyrics. |
| 7 | Deep Rapping: Character Level Neural Models for Automated Rap Lyrics Composition. (2018) | The paper implements a deep learning model that can create lyrics using unsupervised algorithms. So, for this task RNN is used on different architectures and assesses the quality of | Character level language model works by first finding some seed strings in the training corpus. Then each character is tokenized and sorted to know how common it appears. The training data is converted into a vector of integer numbers and is NN is fully trained, an initial seed is picked | The dataset consisted of 4,799 rap lyrics from the original Hip-hop Lyrics Archive. With so much lyrical diversity there's a possibility that it may lower the quality of the generated rap lyrics. So, | The PRNN converged the fastest but it also had the worst cross-entropy loss. GRU was able to outperform both RNN and LSTM when it comes to training quality but at the same time it was the slowest to | Incorporation of rhymes and intelligibility in the algorithm to generate more rhythmic and coherent rap lyrics. . For future works, there can be incorporation of rhymes and intelligibility in the algorithm to |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | cross-entropy loss, rhyme densities score. There's also a Turing test performed to see how close machines can emulate humans. | randomly and converted into a vector of integer numbers. The next character tokens with max probability are predicted and appended into the vector. The character token prediction employs a temperature parameter. The lower the temperature value, the more consecutive rap lyrics will be generated but will have more mistakes. After the integer vector is detokenized, it starts a new set of lyrics then fed to RNN. | the dataset was filtered to concentrate on the songs of 3 artists. When trying different hyper-parameters on a plain rnn, there were 3 hidden layers with 512 neurons each and with a learning rate of 0.0001 which may work best for the task at hand. | train. The lyrics were able to deceive students into thinking if they were real lyrics or not based on the Turing test. A gap identified was that the rhyme density was low. | generate more rhythmic and coherent lyrics. |
| 8 | The Accuracy of Recurrent Neural Networks for Lyric Generation (2018) | The system will consist CNN for image classification and a LSTM neural network trained on a song lyrics data set. | Firstly, an objective test based on repetition and compressibility is performed with varying weights placed in a basic formula-based format. Following this test, a subjective test with human participants is conducted, where we compare similarities and trends between the two results. | The data sets were plain text, unformatted lyrics compiled into a file, with albums averaging a file size of around 20 KB. The data sets were divided at a 1 to 10 ratio for validation compared to training data sets due to their relatively small sizes. | One limitation of this system is the disregard for grammar; not only are the data sets not large enough to provide an adequate data size to learn proper English structure but there is no collaboration with Natural Language Processing (NLP) algorithms or language parsers. With this limitation, the lyrics generated can feel and | Future work includes grammar and NLP development as an area for dramatic improvement of quality of lyrics, as they are currently not present. |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | resemble an artist but ultimately be grammatically incorrect. | |
| 9 | Neural Melody Composition from Lyrics (2018) | Based on RNN,CNN and LSTM | The human annotators are asked to rate a score from 1 to 5 after listening to the songs. Larger scores indicate better quality in all the three metrics. | 18,451 Chinese pop songs, which include melodies with the duration over 800 hours in total, from an online Karaoke app. | A gap identified was that the rhyme density was low. | For future work, plan to incorporate the emotion and the style of lyrics to compose the melody. |
| 10 | Deep Learning in Musical Lyric Generation: An LSTM-Based Approach (2017) | Use an LSTM layer with dropout, a regularization method in which input and recurrent connections to LSTM units are excluded from activation and weight updates to reduce overfitting. | A model was trained on 2000 sampled songs from our dataset. We then created 20 different song "prompts," each of length 16, and fed those same song "prompts" into each of the models trained on each of the genres. | The compiled dataset included 297,876 songs, each labelled one of 14 genres. This complete dataset contained more than 2,000 songs for each genre, with rock having the plurality of songs, with 109,221. | We do observe a decent amount of variation before and after song generation, most notably how our model is unable to capture our I vs. You metric in rock. | Future avenues of work consist of employing different types of models to our song lyric data, most notably simple Markov Models (as those are the simplest way to predict the next word using previous words via transition probabilities) as customization can be done due to the simplistic nature of the model. |
| 11 | Conditional LSTM-GAN for Melody Generation from Lyrics(2018) | A conditional LSTM-GAN model to compose lyrics-conditioned melody where a discriminator can help to ensure that generated melodies have the same distribution as real ones. | The human annotators are asked to rate a score from 1 to 5 after listening to the songs. Larger scores indicate better quality in all the three metrics. | 20,000 pop songs, which include melodies with the duration over 800 hours in total, from an online Karaoke app. | There still are the gaps between melodies generated by our model and ones from human composition, which tells us there is much space we can investigate to improve capability of neural melody generation | i)how to compose melody with the sketch of uncomplete lyrics ii) how to compose polyphonic melody with lyrics. iii) how to predict lyrics when given melody as a condition. |

| | | | | | | |
|---|---|---|---|---|---|---|
| 12 | Automatic Neural Lyrics and Melody Composition (2020) | It uses Automatic Neural Lyrics and Melody Composition ,an attempt to make the whole process of song writing automatic using artificial neural networks. Model trained on a large set of lyric-melody pairs dataset. | Evaluate the generated lyrics and melodies with baseline method and the ground truth human compositions. Separately conduct the subjective evaluation for generated lyrics and lyric-melody pairs. The subjective evaluation is conducted through the Google sheets where Google sheets consists of the samples understudy, radio buttons indicating scores to choose from, and clear instructions on how to rate the samples based on 5 point scale. | The dataset used in our experiments initially created in [9], which come from two different sources: a) the ”LMD-full dataset” of the Lakh MIDI Dataset v0.1 1 and a dataset found on a reddit thread called ”the largest midi collection on the internet. | From the subjective evaluation measures, it was found that there is a gap between the human compositions and generated lyrics and melodies by the proposed model. | There is a lot of scope to explore injecting prior musical knowledge to improve the current model. |
| 13 | GENERATING TURKISH LYRICS WITH LONG SHORT TERM MEMORY (2020) | Based on RNN,CNN and LSTM | Word2Vec is a style which learns word embedding using shallow neural network is used. | The all set contained 932 song lyrics in txt format, as well as basic meta-data including only the lyrics without any other data. In order to use only words, all the punctuation marks and spaces have been erased from the corpus | The experimental results indicates that the generation of lyrics with Turkish songs may be quite difficult because of the Turkish Grammar and sentence structure. | Maybe, line-by-line generation can give better results for a song. |
| 14 | Long Short-Term Memory Recurrent Neural Network Architectures for Generating Music and Japanese Lyrics (2018) | LSTM model that generates Japanese lyrics. | Evaluation is subjective and dependent on how the music sounds like | Char-RNN it was on a dataset of 277KB. | There are multiple areas where the grammar is incorrect, and the sentences do not make any sense. It did learn that the lyric composition is 3-4 lines of text, a line break, another 3-4 lines of text. | To continue working on the implementation to fix the code and improve the performance of the model. There is s understand about LSTM networks both in theory and in practice. There are many optimization techniques involved in neural networks that to further explore. |

| | | | | | | |
|---|---|---|---|---|---|---|
| 15 | Conditional LSTM-GAN for Melody Generation from Lyrics (2019) | A novel deep generative model, conditional Long Short-Term Memory - Generative Adversarial Network (LSTM-GAN) for melody generation from lyrics, which contains a deep LSTM generator and a deep LSTM discriminator both conditioned on lyrics. | Some quantitative measurements are designed to compare the melodies generated by both our proposed model and the baseline, which are shown in the following:<br>• MIDI numbers span: the difference between the highest MIDI number of the sequence and the lowest one.<br>• 3-MIDI number repetitions: a count of how many MIDI numbers 3-grams repetitions occur throughout the sequence | Create a large dataset consisting of 12,197 MIDI songs each with paired lyrics and melody alignment through leveraging different music sources where alignment relationship between syllables and music attributes is extracted | The feedback from subjects indicates that relatively low scores of melody evaluation are generated which might be due to the limited capability of the synthesizer for high pitches | i) how to compose melody with the sketch of uncomplete lyrics<br>ii) how to compose lyrics.<br>iii) how to predict lyrics when given melody as a condition. |
| 16 | Music Generation Based on Convolution-LSTM (2018) | A model that combines Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) for music generation. | First use the accuracy of the prediction to simply evaluate the effect of the model.<br>Sonogram is widely used in the analysis and processing of audio signals. | Downloaded 500 MIDI-format piano songs from midiworld.com as the training data of the model | When training the RNN, the vanishing gradient problem was found and the accuracy was low. | Employing different types of models to our song lyric data |
| 17 | Genre Classification using Feature Extraction and Deep Learning Techniques (2018) | Two Natural Language Processing techniques namely Bag-Of-Words and Term Frequency-Inverse Document Frequency (TFIDF) are used to process the lyrical data. W | Conduct the subjective evaluation for generated lyrics.<br>Larger scores indicate better quality in all the three metrics. | The initial lyrical data is taken from Kaggle [10] and Lyrics Freak. The album artwork and genre labels for each song are downloaded using the Spotify API. The final dataset consists of 10,000 songs belonging to the genres - Christian, Metal, Country, and Rap. | Variation in accuracy, validation accuracy and log loss with change in epochs. | plan to explore more sophisticated Normalized Confusion Matrix Un-normalized Confusion Matrix natural language processing techniques like word2vec, doc2vec and semantic analysis [12] for rigorous cleaning of the lyrical data. Along with it, classifying the songs in an even wider range of genres by using other features like beats per |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | | minute, tone patterns, etc can also have a significant effect on the domain. |

# EXISTING SYSTEM

- Combinatorial approaches have been used to create lyrics before but were predominantly based on the idea of copying the grammar from existing lyrics and then substituting content words with other ones.
- Neural networks have been recently applied to various related tasks. For instance, recurrent neural networks (RNNs) have shown promise in predicting text sequences.

# GAPS IDENTIFIED

- Since a lot of operations were done on the data, the execution time is very large and in accordance to that, the accuracy obtained is not enough.
- The lyrics were able to deceive people into thinking if they were real lyrics or not based on the Turing test.
- Another gap identified was that the rhyme density was low.
- The accuracy is lower in earlier models.

# PROPOSED METHOD

- In this project we will use **Long Short-term memory(LSTM)** that will help the **RNN** to remember about their inputs for a long period of time. Through this program (Lyrics Generator) we will try to generate lyrics for even difficult songs.

- We will train the neural network on lots of words to generate the lyrics of songs based on a given input word or sentence.

- This project can be further extended to identify other forms of music like Jazz, Rock, Pop, Metallic etc. It has other applications like in Auto fillers in industries, Chatbots , Automated email generating system etc.

- Also, we will be using optimizing algorithm known as **Adam Optimizer** is used to reduce the loss and the execution time of the model so as to gain better and faster results.

- We wish to create a generator program which is successfully and opportunely able to generate the desired lyrics of the song with ninety percent accuracy along with bringing melody to the music and building the theme of the song in order to touch the hearts of the public.

# METHODOLOGY

### I) Collecting Data

We downloaded a few datasets from Kaggle and used them to generate the lyrics. The artist specific datasets which we used were of Justin Bieber's, Lorde's, etc. we can even use any dataset according to user specific need.

## DATASET DESCRIPTION

| Singer's Name | The title of a particular song of theirs. | Its Lyrics |
|---|---|---|
| John Mayer 16%<br>Ed Sheeran 14%<br>Other (243) 70% | 347<br>unique values | 347<br>unique values |
| Phoebe Bridgers | Motion Sickness | I hate you for what you did And I miss you like a little kid I faked it every time But that's alrig... |
| Phoebe Bridgers | Killer | Sometimes I think I'm a killer I scared you in your house I even scared myself by talking About Dah... |
| Phoebe Bridgers | Georgia | Georgia, Georgia, I love your son And when he gets older, he might be the one He might be the one G... |

### II) Cleaning the Data

The raw data collected first needs to be processed before we can train it. We remove all the errors of white spaces and commas. We also make a list of all the

unique words that were used, and convert it to an array. Then, we encode all of them using the one-hot-encoding method. There are several other methods of doing this, like creating dummy variables, but this way of encoding gives the highest efficiency of them all.

**III) Defining the Model Class**

We define a class for our Recurrent Neural Network, containing methods to build the basic structure of our model. This includes, the input and output layers, the nonlinear function applied to the input layer and hidden layers, the forward propagation algorithm to get the final output and the linear function used in the transition from last hidden layer to the output layer. We also define the Gated Recurrent Units to keep track of what words have we used up till this point. This better predicts the next words to come.

**IV) Training the Data**

We use the array of words to find the probability that one word follows another. These probabilities are used as our training dataset. And it also generates the weights for our neural network. These weights are tuned to minimise our Loss function to give the optimum values. Once this is done, we are ready to test our model on new training set data

# INNOVATION COMPONENT IN THE PROJECT

The innovative part of this project is the use of LSTM (Long Short Term Memory). There are several reasons for this.

- This becomes an important factor when we use recurrent neural networks as all our previous outputs are being stored. The LSTM is used when the sequence is long.
- However, this does not apply to our code where we use a sequence length of one. This means that, first we predict what letters generally come after others, and then, after a few iterations (epochs) we predict which words follow others.
- This is the innovative part of our project, as most codes implement lyrics generators using LSTMs.

## COMPONENTS AND TOOLS USED:

### *SOFTWARE COMPONENTS:*

- Python IDE
- Jupyter Notebook
- Torch
- Kaggle
- LSTM Model

### *HARDWARE COMPONENTS:*

- A Windows/Linux based system
- 4GB RAM
- I3 Processor

### *TOOLS USED:*

- Anaconda / Jupyter Notebook
- Kaggle
- Google Colaboratory

# IMPLEMENTATION

## Step 1:

Importing all necessary libraries.

```
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout, Bidirectional
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam
from tensorflow.keras import regularizers
import tensorflow.keras.utils as ku
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
```

## Step 2:

Loading our (.txt) dataset file and displaying some sample lyrics for checking correct import of the dataset.

```
[2] # importing dataset in .txt format
    # loading the dataset file and display lyrics
    lyric=open('lorde.txt').read()
    print(lyric[:2000])
```

```
Well, summer slipped us underneath her tongue
Our days and nights are perfumed with obsession
Half of my wardrobe is on your bedroom floor
Use our eyes, throw our hands overboard
I am your sweetheart psychopathic crush
Drink up your movements, still I can't get enough
I overthink your punctuation use
Not my fault, just a thing that my mind do
A rush at the beginning
I get caught up, just for a minute
But lover, you're the one to blame
All that you're doing
Can you hear the violence?
Megaphone to my chest
Broadcast the boom, boom, boom, boom
And make 'em all dance to it
```

## Step 3:

Since the word carries capital letters as well, changing it to lowercase. Also keeping dataset as per line will be more appropriate as it will learn the sentences formed to yield better performance.

```
# pre-processing datset
# converting into lowercase and split the datset
corpus=lyric.lower().split('\n')
for i in range(40,60):
 print(corpus[i])
```

```
but lover, you're the one to blame
all that you're doing
can you hear the violence?
megaphone to my chest
broadcast the boom, boom, boom, boom
and make 'em all dance to it
broadcast the boom, boom, boom, boom
and make 'em all dance to it
broadcast the boom, boom, boom, boom
and make 'em all dance to it
broadcast the boom, boom, boom, boom
and make 'em all dance to it
broadcast the boom, boom, boom, boom
and make 'em all dance to it
our thing progresses
i call and you come through
blow all my friendships
to sit in hell with you
but we're the greatest
they'll hang us in the louvre
```

## Step 4:

Tokenizer creates the tokens for each line present in the corpus and measuring the number of the tokens created.

```
[4] # tokenizing
    tokenizer = Tokenizer()
    tokenizer.fit_on_texts(corpus)
    total_words = len(tokenizer.word_index) + 1
    total_words
    word_index = tokenizer.word_index
    print(word_index)
```

```
{'boom': 1, 'the': 2, 'to': 3, 'and': 4, 'all': 5, 'my': 6, 'i': 7, 'broadcast': 8, 'make': 9, "'em": 1
```

## Step 5:

Creating input sequences using list of tokens. Here we are generating tokens for each word and its preceding word for each line.

```
[5] # using list of tokens create input sequences
    input_sequences = []
    for line in corpus:
     token_list = tokenizer.texts_to_sequences([line])[0]

     for i in range(1, len(token_list)):
         n_gram_sequence = token_list[:i+1]
         input_sequences.append(n_gram_sequence)
    for i in range(20):
     print(input_sequences[i])
```

```
[28, 29]
[28, 29, 30]
[28, 29, 30, 17]
[28, 29, 30, 17, 31]
[28, 29, 30, 17, 31, 32]
[28, 29, 30, 17, 31, 32, 33]
[13, 34]
[13, 34, 4]
[13, 34, 4, 35]
[13, 34, 4, 35, 36]
[13, 34, 4, 35, 36, 37]
[13, 34, 4, 35, 36, 37, 18]
[13, 34, 4, 35, 36, 37, 18, 38]
[39, 40]
[39, 40, 6]
[39, 40, 6, 41]
[39, 40, 6, 41, 42]
[39, 40, 6, 41, 42, 43]
[39, 40, 6, 41, 42, 43, 14]
[39, 40, 6, 41, 42, 43, 14, 44]
```

## Step 6:

Since the length of the arrays formed is different hence padding of length of the longest array is required in order to make the array length uniform.

```
[6] # padding either post-padding or pre-padding
    max_sequence_len = max([len(x) for x in input_sequences])
    input_sequences = np.array(pad_sequences(input_sequences, maxlen=max_sequence_len,
    padding='pre'))
    print(max_sequence_len)
    print(input_sequences)
```

```
10
[[ 0  0  0 ...  0 28 29]
 [ 0  0  0 ... 28 29 30]
 [ 0  0  0 ... 29 30 17]
 ...
 [ 0  0  0 ... 91 17 27]
 [ 0  0  0 ... 17 27  2]
 [ 0  0  0 ... 27  2 92]]
```

## Step 7:

As we have formed our data into array form, now we can build a model to process the same.

```
# Model building
model = Sequential()
model.add(Embedding(1372, 160, input_length=max_sequence_len-1))
model.add(Bidirectional(LSTM(200, return_sequences = True)))
model.add(Dropout(0.2))
model.add(LSTM(100))
model.add(Dense(1372/2, activation='relu', kernel_regularizer=regularizers.l2(0.001)))
model.add(Dense(93, activation='softmax'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
print(model.summary())
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 9, 160)            219520
_________________________________________________________________
bidirectional (Bidirectional (None, 9, 400)            577600
_________________________________________________________________
dropout (Dropout)            (None, 9, 400)            0
_________________________________________________________________
lstm_1 (LSTM)                (None, 100)               200400
_________________________________________________________________
dense (Dense)                (None, 686)               69286
_________________________________________________________________
dense_1 (Dense)              (None, 93)                63891
=================================================================
Total params: 1,130,697
Trainable params: 1,130,697
Non-trainable params: 0
_________________________________________________________________
None
```

## Step 8:

Training the Model :

```
# create predictors and label
predictors, label = input_sequences[:,:-1],input_sequences[:,-1]
label = ku.to_categorical(label, num_classes=total_words)
```

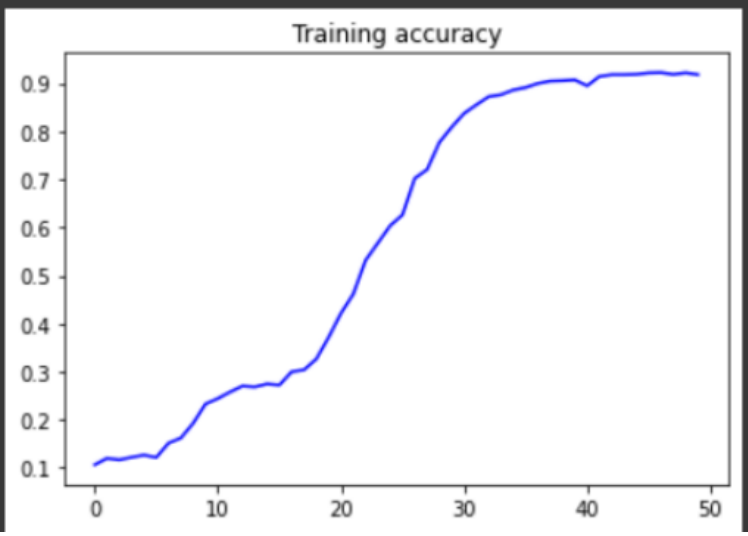
```
# model training
history = model.fit(predictors, label, epochs=50, verbose=1)
```

```
Epoch 1/50
178/178 [==============================] - 22s 91ms/step - loss: 0.2789 - accuracy: 0.0773
Epoch 2/50
178/178 [==============================] - 16s 90ms/step - loss: 0.0580 - accuracy: 0.1262
Epoch 3/50
178/178 [==============================] - 16s 92ms/step - loss: 0.0552 - accuracy: 0.1111
Epoch 4/50
178/178 [==============================] - 16s 91ms/step - loss: 0.0550 - accuracy: 0.1144
Epoch 5/50
178/178 [==============================] - 16s 91ms/step - loss: 0.0545 - accuracy: 0.1258
Epoch 6/50
178/178 [==============================] - 16s 92ms/step - loss: 0.0542 - accuracy: 0.1150
Epoch 7/50
178/178 [==============================] - 17s 94ms/step - loss: 0.0511 - accuracy: 0.1558
Epoch 8/50
178/178 [==============================] - 17s 93ms/step - loss: 0.0492 - accuracy: 0.1588
Epoch 9/50
178/178 [==============================] - 16s 92ms/step - loss: 0.0475 - accuracy: 0.1808
Epoch 10/50
178/178 [==============================] - 16s 93ms/step - loss: 0.0461 - accuracy: 0.2263
Epoch 11/50
178/178 [==============================] - 17s 93ms/step - loss: 0.0438 - accuracy: 0.2400
Epoch 12/50
178/178 [==============================] - 17s 97ms/step - loss: 0.0423 - accuracy: 0.2560
Epoch 13/50
```

## Step 9:

Analysing the results: By plotting the training accuracy and training loss, we can infer the model performance using MatPlotLib.

```
# result analysis
acc = history.history['accuracy']
loss = history.history['loss']
epochs = range(len(acc))
plt.plot(epochs, acc, 'b', label='Training accuracy')
plt.title('Training accuracy')
plt.figure()
plt.plot(epochs, loss, 'b', label='Training Loss')
plt.title('Training loss')
plt.legend()
plt.show()
```




# RESULT AND DISCUSSIONS

## Testing the Model-

▾ 7. Testing the model

Take the input from user, converting into padded array and printing results

```
[ ]  next_words = 20
     seed_text = "watch the wasters blow the speakers "
```

```
[ ]
     for _ in range(next_words):
       token_list = tokenizer.texts_to_sequences([seed_text])[0]
       token_list = pad_sequences([token_list], maxlen=max_sequence_len-1, padding='pre')
       predicted = model.predict_classes(token_list, verbose=0)
       output_word = ""
       for word, index in tokenizer.word_index.items():
         if index == predicted:
           output_word = word
           break
       seed_text += " " + output_word
     print(seed_text)
```

```
watch the wasters blow the speakers  lead the party let the people know all the lovers but people talk to me i'm slipping out of reach
```

```
[ ] # trying random samples:

    next_words = 10
    seed_text = "ive never seen a diamond in the flesh i cut my"

    for _ in range(next_words):
      token_list = tokenizer.texts_to_sequences([seed_text])[0]
      token_list = pad_sequences([token_list], maxlen=max_sequence_len-1, padding='pre')
      predicted = model.predict_classes(token_list, verbose=0)
      output_word = ""
      for word, index in tokenizer.word_index.items():
        if index == predicted:
          output_word = word
          break
      seed_text += " " + output_word
    print(seed_text)
```

```
ive never seen a diamond in the flesh i cut my teeth on wedding rings in the movies that we watched
```

- So basically through this project we discussed about the use and advantages of Long Short-term memory that will help the RNN to remember about their inputs for a long period of time.

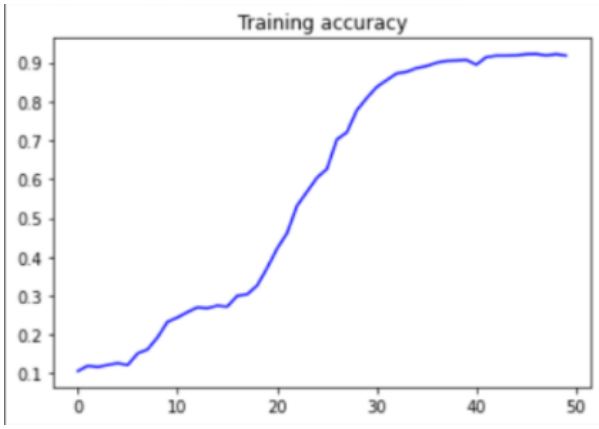
- Through this program (Lyrics Generator) we also tried to generate lyrics for even difficult songs. We will train the neural network on lots of words to generate the lyrics of songs based on a given input word or sentence.

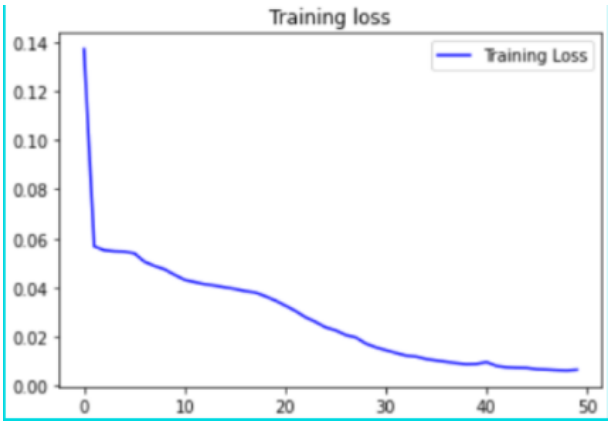
- At the end of this project we have tried to develop a program with as much high accuracy as possible in order to be a successful product.

- As a result we can see the induced lyrics from line one, completes the sentence of a particular song and the next one is carried out to another song because after the one line induced lyrics from the first song is predicted, the prediction gets distorted because it was made of a single word being used as a corpus.

**TRAINING ACCURACY AND LOSS :**

The results show that the accuracy of our model **increases** with training, and the loss factor **decreases** which increases the efficiency of our model drastically over time.




# CONCLUSION

- The lyrics generation was successfully implemented using Recursive Neural Network along with LSTM. The dataset taken was enough to train the model and generate a good quality of rap lyrics.

- The accuracy of the model was increased by the use of Adam Optimizer which also reduced the loss caused during the training. A self-generated rap was created using this model.

# FUTURE SCOPE

- The future scope of this model could involve providing it with a web interface that makes it easier for the artist to generate the required lyrics.

- Also, a melody can be designed using the generated lyrics by making use of some packages from python library.

- Another scope can include generating paragraph wise lyrics so that the artist can add various themes to different verses in his lyrics.

# REFERENCES


*1. E. Malmi, P. Takala, H. Toivonen, T. Raiko, and A. Gionis, "Dopelearning: A computational approach to rap lyrics generation," in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 195–204, 2016.*

*2. P. Potash, A. Romanov, and A. Rumshisky, "Ghostwriter: Using an lstm for automatic rap lyric generation," in Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 1919–1924, 2015.*

*3. O. Vechtomova, H. Bahuleyan, A. Ghabussi, and V. John, "Generating lyrics with variational autoencoder and multi-modal artist embeddings," arXiv preprint arXiv:1812.08318, 2018.*

*4. Y. Fan, X. Lu, D. Li, and Y. Liu, "Video-based emotion recognition using cnn-rnn and c3d hybrid networks," in Proceedings of the 18th ACM International Conference on Multimodal Interaction, pp. 445–450, 2016.*

*5. M. Loller-Andersen and B. Gamb¨ack, "Deep learning-based poetry generation given visual input.," in ICCC, pp. 240–247, 2018.*

*6. A. C. T. Fernandez, K. J. M. Tarnate, and M. Devaraj, "Deep rapping: Character level neural models for automated rap lyrics composition," International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN, pp. 2278–3075, 2018.*

*7. A. Lambert, T. Weyde, and N. Armstrong, "Perceiving and predicting expressive rhythm with recurrent neural networks," in Proceedings of the 12th International Conference in Sound and Music Computing, SMC 2015, SMC15,2015.*

*8. B. F. Bhaukaurally, M. H. A. Didorally, and S. Pudaruth, "A semi-automated lyrics generation tool for mauritian sega," IAES International Journal of Artificial Intelligence (IJ-AI), vol. 1, no. 4, 2012.*