Lyric generator using LSTM'S

SANAGALA RISHI PREETHAM (17BEE0043), B. NIKHIL (17BEE0068)

UDATHA SANDEEP (17BEE00259)

Electrical and Electronics Engineering

VIT University Vellore, Tamil Nadu

Abstract:

Lyric generator is a model when trained on a dataset being able to generate the song of the lyrics given as an input and also generates meaningful text without being explicitly programmed to do so. The model is trained by the datasets of the lyrics of various songs of various famous singers. By providing the initial lyrics of the song as input it generates the lyrics of the whole song. This model is developed using **Long Short Term Memory (LSTM)**. LSTM is an Architecture of **Recurrent Neural Networks.** These are the widely used Neural networks in speech detection, Text Generation and classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series. Therefore it is well suited to learn from important experiences that have very long time lags in between. It has been found that the LSTM Architecture performs better and shows an accuracy of 98%. This kind of model is going to produce a meaningful text so that it can be widely used in many applications like Text suggestions on keyboards, Automated email generation systems, chat box, Text suggestions on keypad, chat boxes etc.

KEYWORDS: Long Short Term Memory (LSTM), Recurrent Neural Networks, Lyric Generator, keras, python, numpy etc.

1 INTRODUCTION:

Lyric Generator is a Lyric generating model capable of generating lyrics of a actual song when trained on a dataset being able to generate the lyrics of the song for which input is given as initial lyrics of the song and also capable in generating meaningful text which is done by using LSTM because **Long Short-Term Memory** (LSTM) networks are an extension for recurrent neural networks, which basically extends their memory. LSTM is an Architecture of **Recurrent Neural Networks (RNN).** Recurrent neural networks can also be used as generative models. This means that in addition to being used for predictive models (making predictions) they can learn the sequences of a problem and then generate entirely new plausible sequences for the problem domain. Generative models like this are useful not only to study how well a model has learned a problem, but to learn more about the problem domain itself. Datasets are given as Lyrics of the songs, Vocabulary, books etc. The programing of the model is going to be implemented in Python by using Keras Libraries. Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow. Data is going to be optimised and pre processed before fitting into model. The model is going to be available in the form of a website which asks the user to provide the initial lyrics of the song as an input and displays the lyrics of the whole song on the screen.

2 LITERATURE REVIEW:

2.1 **Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network**

Author: Alex Sherstinsky

LSTM networks have received a wealth of coverage in scientific journals, technical blogs, and implementation guides. However, in most articles, the inference formulas for the LSTM network and its parent, RNN, are stated axiomatically, while the training formulas are omitted altogether. In addition, the technique of “unrolling” an RNN is routinely presented without justification throughout the literature. The goal of this paper is to explain the essential RNN and LSTM fundamentals in a single document. Drawing from concepts in signal processing, we formally derive the canonical RNN formulation from differential equations. We then propose and prove a precise statement, which yields the RNN unrolling technique. We also review the difficulties with training the standard RNN and address them by transforming the RNN into the “Vanilla LSTM”1 network through a series of logical arguments. We provide all equations pertaining to the LSTM system together with detailed descriptions of its constituent entities. Albeit unconventional,

our choice of notation and the method for presenting the LSTM system emphasizes ease of understanding.

2.2 Long short term memory(Lstm)

Authors: Sepp Hochreiter and Jürgen Schmidhuber

A common LSTM unit is composed of a **cell**, an **input gate**, an **output gate** and a **forget gate**. The cell remembers values over arbitrary time intervals and the three *gates* regulate the flow of information into and out of the cell.

LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series. LSTMs were developed to deal with the exploding and vanishing gradient problems that can be encountered when training traditional RNNs. Relative insensitivity to gap length is an advantage of LSTM over RNNs, hidden Markov models and other sequence learning methods in numerous applications.

2.3 Semantically Conditioned LSTM-based Natural Language Generation for Spoken Dialogue Systems

Authors: Tsung-Hsien Wen, Milica Gasiˇ c, Nikola Mrk , Pei-Hao Su, David Vandyke and Steve Young .

The authors presents a statistical language generator based on a semantically controlled Long Short-term Memory (LSTM) structure. The LSTM generator can learn from unaligned data by jointly optimising sentence planning and surface realization using a simple cross entropy training criterion, and language variation can be easily achieved by sampling from output candidates. With fewer heuristics, an objective evaluation in two differing test domains showed the proposed method improved performance compared to previous methods. Human judges scored the LSTM system higher on informativeness and naturalness and overall preferred it to the other systems.

2.4Text-based LSTM networks for Automatic Music Composition

Authors: Keunwoo Choi, George Fazekas, Mark Sandler

According to author Music can be represented as a sequence of events and thus it can be modelled as conditional probabilities between musical events. For example, in harmonic tracks, some chords are more likely to occur than others given the previous chords, while the whole chord progressions often depend on the global key of the music. In many automatic composition systems, these relationships are simplified by assuming that the probability of the current state p (n) only depends on the probabilities of the states in the past p (n − k)...p (n − 1). A sequence of musical events - notes, chords, rhythm patterns - is

generated by predicting the following event given a seed sequence. We introduced an algorithm of text-based LSTM networks for automatic composition and reported results for generating chord progressions and rock drum tracks. Word-RNNs showed good results in both cases while char-RNNs only successfully learned chord progressions. The experiments show LSTM provides a way to learn the sequence of musical events even when the data is given as text. Text Generation Using R**ecurrent** Neural Networks

Text generation is a popular problem in Data Science and Machine Learning, and it is a suitable task for Recurrent Neural Nets. Author uses TensorFlow to build an RNN text generator and builds a high-level API in Python3. Creates a recurrent neural network with a TensorFlow RNN cell (which performs dynamic unrolling of the inputs). It has an output projection layer which produces the final probability for each character class. It generates the text by sampling the next character based on the probability distribution of the last character of the current sequence.

2.5 Neural networks used for speech recognition

Authors: **Wouter Gevaert**, Georgi Tsenov

This authors are explaining that neural networks can be very powerful speech signal classifiers. A small set of words could be recognized with some very simplified models. The pre-processing quality is giving the biggest impact on the neural networks performance. In some cases where the spectrogram combined with entropy based endpoint detection is used we observed poor classification performance results, making this combination as a poor strategy for the pre-processing stage. On the other hand we observed that Mel Frequency Ceptstrum Coefficients are a very reliable tool for the pre-processing stage, with the good results they provide. Both the Multilayer Feedforward Network with backpropagation algorithm and the Radial Basis Functions Neural Network are achieving satisfying results when Mel Frequency Ceptstrum Coefficients are used.

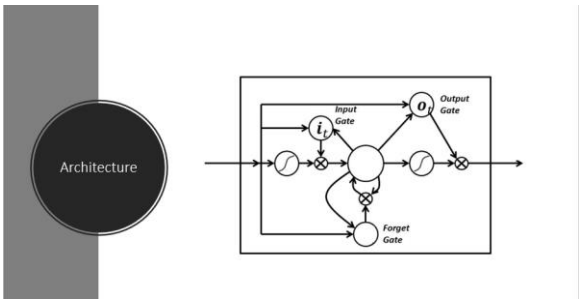
2.6 Semantically Conditioned LSTM-based Natural Language Generation for Spoken Dialogue Systems

Authors: Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, Steve Young

Natural language generation (NLG) is a critical component of spoken dialogue and it has a significant impact both on usability and perceived quality. Most NLG systems in common use employ rules and heuristics and tend to generate rigid and stylised responses without the natural variation of human language. They are also not easily

scaled to systems covering multiple domains and languages. This paper presents a statistical language generator based on a semantically controlled Long Short-term Memory (LSTM) structure.

3. Architecture



Long short-term memory (**LSTM**) units (or blocks) are a building unit for layers of a recurrent neural network (RNN). A RNN composed of LSTM units is often called an LSTM network. A common LSTM unit is composed of a **cell**, an **input gate**, an **output gate** and a **forget gate.**

LSTM network have a sequence like structure, but the recurring network has a different module. Instead of having single neural network layer, they have small parts connected to each other which function in storing and removal of memory.

4. ALGORITHM

4.1 Recurrent Neural Networks:

Recurrent Neural Networks is one of the most well-known Neural Networks utilized in Natural Language Processing as a result of its promising outcomes. The uses of RNN in language models comprise of two principle draws near. We can either cause the model to anticipate or figure the sentences for us and right the blunder during forecast or we can prepare the model on the specific class and it can deliver content like it, which is interesting.

The rationale behind an RNN is to think about the succession of the info. For us to foresee the following word in the sentence we have to recollect what word showed up in the past time step. These neural systems are called Recurrent on the grounds that this progression is done for each info. As these neural systems think about the last word during foreseeing, it acts like a memory stockpiling unit that stores it for a brief timeframe.

The thought is to prepare the RNN with numerous successions of words and the objective next word. As a disentangled model, on the off chance that each sentence is a rundown of five words, at that point, the objective is a rundown of just a single component.

We don't really send the strings, however a vectorized portrayal of the word inside a lexicon of potential words (more on that later). The thought is that after numerous ages the RNN will learn "the style" of how the corpus is composed, attempting to modify loads of the system to foresee the following word given an arrangement of the N past words.

4.2 Long Short Term Memory

The initial phase in the LSTM is to choose which data to be discarded in from the cell in that specific time step. It is chosen by the sigmoid capacity which precludes in the event that it is 0 and stores on the off chance that it is 1. It takes a gander at the past state ht-1 and the present info xt and registers the capacity. To comprehend the enactment capacities and the math behind it go here.

At that point, we have another layer which comprises two sections. One is the sigmoid capacity and the other is the tanh. In the sigmoid capacity, it chose which esteems to let through(0 or 1). Also, in the tanh work it gives the weight to the qualities which are passed choosing their degree of significance(- 1 to 1).

At last, we have to choose what we're going to yield. This yield will be founded on our cell state, however, it will be a sifted rendition. In the first place, we run a sigmoid layer that chooses what parts of the cell state we're going to yield. At that point, we put the cell state through tanh to push the qualities to be between - 1 and 1 and duplicate it by the yield of the sigmoid door, so we just yield the parts we chose to do.

5. Model construction

The model is built by using various libraries numpy, keras and pandas etc and undergoes cleaning data, one hot coding and vectorization and then takes the input as lyrics of the song and generates whole lyrics as text.

An Online platform is developed by us such that you can use this model through the website.

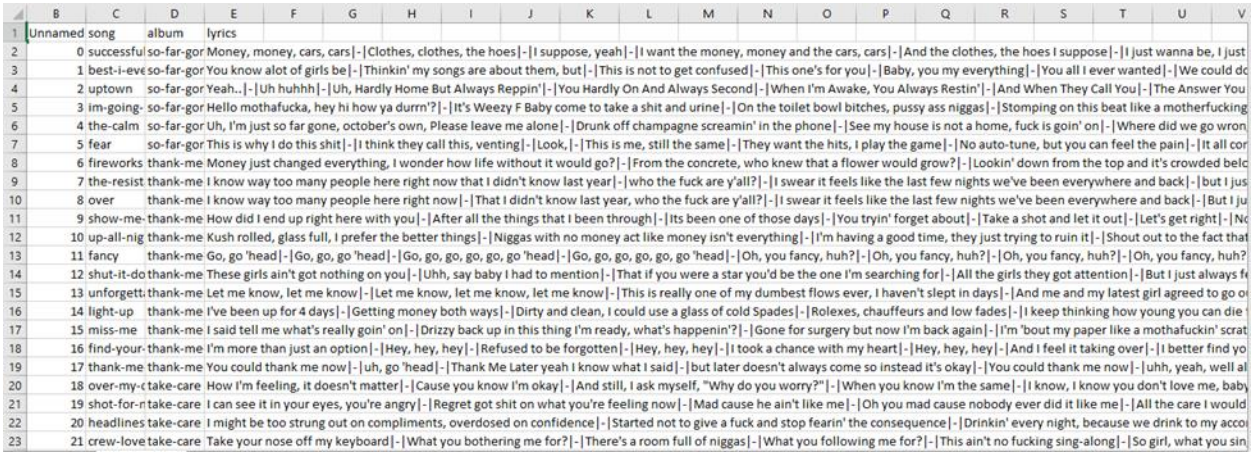
Datasets used are the lyrics of the songs of various popular singers like Drake, Eminem and Usher.

5.1 Code Implementation for Lyric generator:

```
#ImportingModul
es
                    import numpy as np
                    import pandas as pd
                    import re
                    from keras.models
                    import Sequential
                    from keras.layers
                    import Dropout
                    from keras.callbacks
                    import ModelCheckpoint
                    from keras.layers
                    import Dense,
                    Activation,
                    Bidirectional
                    from keras.layers
                    import
                    CuDNNLSTM,GlobalMaxPoo
                    l1D
```

%The modules are being imported in the above shown lines of code. All the required modules for the code are given.

Data sets of lyrics



	Unnamed	song	album	lyrics
2	0	successful	so-far-gor	Money, money, cars, cars|-|Clothes, clothes, the hoes|-|I suppose, yeah|-|I want the money, money and the cars, cars|-|And the clothes, the hoes I suppose|-|I just wanna be, I just
3	1	best-i-eve	so-far-gor	You know alot of girls be|-|Thinkin' my songs are about them, but|-|This is not to get confused|-|This one's for you|-|Baby, you my everything|-|You all I ever wanted|-|We could do
4	2	uptown	so-far-gor	Yeah..|-|Uh huhhh|-|Uh, Hardly Home But Always Reppin'|-|You Hardly On And Always Second|-|When I'm Awake, You Always Restin'|-|And When They Call You|-|The Answer You
5	3	im-going-	so-far-gor	Hello mothafucka, hey hi how ya durrn'?|-|It's Weezy F Baby come to take a shit and urine|-|On the toilet bowl bitches, pussy ass niggas|-|Stomping on this beat like a motherfucking
6	4	the-calm	so-far-gor	Uh, I'm just so far gone, october's own, Please leave me alone|-|Drunk off champagne screamin' in the phone|-|See my house is not a home, fuck is goin' on|-|Where did we go wron
7	5	fear	so-far-gor	This is why I do this shit|-|I think they call this, venting|-|Look,|-|This is me, still the same|-|They want the hits, I play the game|-|No auto-tune, but you can feel the pain|-|It all cor
8	6	fireworks	thank-me	Money just changed everything, I wonder how life without it would go?|-|From the concrete, who knew that a flower would grow?|-|Lookin' down from the top and it's crowded belo
9	7	the-resist	thank-me	I know way too many people here right now that I didn't know last year|-|who the fuck are y'all?|-|I swear it feels like the last few nights we've been everywhere and back|-|but I jus
10	8	over	thank-me	I know way too many people here right now|-|That I didn't know last year, who the fuck are y'all?|-|I swear it feels like the last few nights we've been everywhere and back|-|But I ju
11	9	show-me-	thank-me	How did I end up right here with you|-|After all the things that I been through|-|Its been one of those days|-|You tryin' forget about|-|Take a shot and let it out|-|Let's get right|-|No
12	10	up-all-nig	thank-me	Kush rolled, glass full, I prefer the better things|-|Niggas with no money act like money isn't everything|-|I'm having a good time, they just trying to ruin it|-|Shout out to the fact that
13	11	fancy	thank-me	Go, go 'head|-|Go, go, go 'head|-|Go, go, go, go, go, go 'head|-|Go, go, go, go, go, go 'head|-|Oh, you fancy, huh?|-|Oh, you fancy, huh?|-|Oh, you fancy, huh?|-|Oh, you fancy, huh?
14	12	shut-it-do	thank-me	These girls ain't got nothing on you|-|Uhh, say baby I had to mention|-|That if you were a star you'd be the one I'm searching for|-|All the girls they got attention|-|But I just always fe
15	13	unforgett	thank-me	Let me know, let me know|-|Let me know, let me know, let me know|-|This is really one of my dumbest flows ever, I haven't slept in days|-|And me and my latest girl agreed to go o
16	14	light-up	thank-me	I've been up for 4 days|-|Getting money both ways|-|Dirty and clean, I could use a glass of cold Spades|-|Rolexes, chauffeurs and low fades|-|I keep thinking how young you can die
17	15	miss-me	thank-me	I said tell me what's really goin' on|-|Drizzy back up in this thing I'm ready, what's happenin'?|-|Gone for surgery but now I'm back again|-|I'm 'bout my paper like a mothafuckin' scrat
18	16	find-your-	thank-me	I'm more than just an option|-|Hey, hey, hey|-|Refused to be forgotten|-|Hey, hey, hey|-|I took a chance with my heart|-|Hey, hey, hey|-|And I feel it taking over|-|I better find yo
19	17	thank-me	thank-me	You could thank me now|-|uh, go 'head|-|Thank Me Later yeah I know what I said|-|but later doesn't always come so instead it's okay|-|You could thank me now|-|uhh, yeah, well al
20	18	over-my-(	take-care	How I'm feeling, it doesn't matter|-|Cause you know I'm okay|-|And still, I ask myself, "Why do you worry?"|-|When you know I'm the same|-|I know, I know you don't love me, baby
21	19	shot-for-r	take-care	I can see it in your eyes, you're angry|-|Regret got shit on what you're feeling now|-|Mad cause he ain't like me|-|Oh you mad cause nobody ever did it like me|-|All the care I would
22	20	headlines	take-care	I might be too strung out on compliments, overdosed on confidence|-|Started not to give a fuck and stop fearin' the consequence|-|Drinkin' every night, because we drink to my acco
23	21	crew-love	take-care	Take your nose off my keyboard|-|What you bothering me for?|-|There's a room full of niggas|-|What you following me for?|-|This ain't no fucking sing-along|-|So girl, what you sin

Datasets

```
#ImportingData
                    data =
                    pd.read_csv('drake-
                    songs.csv')


                    text = ''
```

%Here, the dataset which is required for training of the program is imported from the system in which the it is saved.

```
#CleaningDat
a
                for index, row in
                data['lyrics'].iteritems(
                ):
                    cleaned =
                str(row).lower().replace(
                ' ', '\n').replace('|-
                |','\n')
                    text = text + "
                ".join(re.findall(r"[a-
                z']+", cleaned))
```

```
tokens = re.findall(r"[a-
z'\s]", text)


chars =
sorted(list(set(tokens)))
char_indices = dict((c,
i) for i, c in
enumerate(chars))
indices_char = dict((i,
c) for i, c in
enumerate(chars))


vocab_size = len(chars)
```

%In the above given lines of the code, the data is cleaned. All the spaces (‘ ‘) are replaced new line. All the hyphens (‘ – ‘) are also replaced by new lines. All the upper case letters are converted to lower case alphabets.

```
#Vectoriz
ing
            maxlen = 40
            step = 1
            sentences = []
            next_char = []
            for i in range(0,
            len(text)-maxlen, step):

            sentences.append(text[i:i+m
            axlen])

            next_char.append(text[i+max
            len])
```

%The concept of vectorizing is shown in the above lines of the code. Vectorization is used to improve the vocabulary of machine or the program.

```
#OneHotEncodi
ng
                x =
                np.zeros((len(sentences)
                , maxlen, len(chars)),
                dtype = np.bool)
                y =
                np.zeros((len(sentences)
                , len(chars)))

                for i, sentence in
                enumerate(sentences):
                  for j, char in
                enumerate(sentence):
                    x[i, j,
                char_indices[char]] = 1
                  y[i,
                char_indices[next_char[i
                ]]] = 1
```

%One hot encoding: This is where the integer encoded variable is removed and a new binary variable is added for each unique integer value.

```
#ModelCos
truction
            model = Sequential()
            model.add(Bidirectional(CuD
            NNLSTM(256,input_shape =
            (maxlen,vocab_size),return_
            sequences = True)))
            model.add(Dropout(0.2))
            model.add(Bidirectional(CuD
            NNLSTM(128,
            return_sequences = True)))
            model.add(Dropout(0.2))
            model.add(Bidirectional(CuD
            NNLSTM(64, return_sequences
            = True)))
            model.add(Dropout(0.2))
            model.add(GlobalMaxPool1D()
            )
            model.add(Dense(len(chars))
            )
            model.add(Activation('softm
            ax'))
            model.compile(loss='categor
            ical_crossentropy',
            optimizer = "nadam",metrics
            = ['accuracy'])
```

%The concept of LSTM is applied in the given lines of code. The first layer of the model is provided with 256 cells and is subsequently reduced to half within three layers.

5.2 Code for the web application

```
# -
*-
cod
ing
:
utf
-8
-*-
    """
    Created on Tue Oct 23 19:16:30
    2019

    @author: tanma
    """
    # Importing Modules
    import numpy as np
    import pandas as pd
    import tensorflow as tf
    import random
    import re
    import flask
    import os.path
    from flask_cors import CORS
    from flask import request
    from load import *
    from keras.models import
    load_model
    from flask import Response




    # Importing Data
    data = pd.read_csv('drake-
    songs.csv')


    text = ''


    # Cleaning Data
    for index, row in
    data['lyrics'].iteritems():
        cleaned =
    str(row).lower().replace(' ',
    '\n').replace('|-|','\n')
        text = text + "
    ".join(re.findall(r"[a-z']+",
    cleaned))

    tokens = re.findall(r"[a-z'\s]",
    text)



    chars = sorted(list(set(tokens)))
    char_indices = dict((c, i) for i,
    c in enumerate(chars))
    indices_char = dict((i, c) for i,
    c in enumerate(chars))


    vocab_size = len(chars)


    # Vectorizing
    maxlen = 40
    step = 1
    sentences = []
    next_char = []
    for i in range(0, len(text)-
    maxlen, step):

    sentences.append(text[i:i+maxlen])
      next_char.append(text[i+maxlen])


    # One Hot Encoding
    x = np.zeros((len(sentences),
    maxlen, len(chars)), dtype =
    np.bool)
    y = np.zeros((len(sentences),
    len(chars)))
```

```
for i, sentence in
enumerate(sentences):
  for j, char in
enumerate(sentence):
    x[i, j, char_indices[char]] =
1
  y[i, char_indices[next_char[i]]]
= 1


# Predicting
global model,graph
model,graph = init()




app = flask.Flask(__name__)


CORS(app)


@app.route("/",
methods=["GET","POST"])
def predict():
    generated = ''
    with graph.as_default():

        start_index=random.randint(
0,len(text)-maxlen-1)

        sent=text[start_index:start
_index+maxlen]
                generated+=sent
                for i in
range(200):

x_sample=generated[i:i+maxlen]

x=np.zeros((1,maxlen,vocab_size))
                    for j in
range(maxlen):

x[0,j,char_indices[x_sample[j]]]=1

probs=model.predict(x)

probs=np.reshape(probs,probs.shape
[1])

ix=np.random.choice(range(vocab_si
ze),p=probs.ravel())

generated+=indices_char[ix]
    print('Returning
prediction...')
    return '<!DOCTYPE html><html
lang="en"><head><title>Drake Song
Generator</title><link
href="https://stackpath.bootstrapc
dn.com/bootstrap/4.1.3/css/bootstr
ap.min.css"
rel="stylesheet"></head><body><div
class="jumbotron"><h1
class="display-4">Drake Lyric
Generator</h1><p class="lead">This
is an Deep-Learning Model to
generate Lyrics of the Famous
Rapper Drake.</p><hr class="my-
4"><p>It uses LSTM for time series
prediction.</p><p>Tanmay
Thakur</p><p>Rushi
Varun</p></hr></div><div
class="jumbotron"><h3
class="display-5">Start
Text:</h3><div
class="container"><p
class="lead">'+sent+'</p></div><h3
class="display-5">Generated
Text:</h3><div
class="container"><p
class="lead">'+generated+'</p></di
v></div></body></html>'
```

```
if __name__ == '__main__':
        port =
int(os.environ.get("PORT", 5000))
        print('Starting app...')
        app.run(host='0.0.0.0',
port=port)

def custom_predict(sent):
    generated = ''
    with graph.as_default():
        while len(sent) < maxlen:
                        sent+=' '
        generated+=sent.lower()
        for i in range(200):

x_sample=generated[i:i+maxlen]

x=np.zeros((1,maxlen,vocab_size))
                                    for j in
range(maxlen):

x[0,j,char_indices[x_sample[j]]]=1

probs=model.predict(x)

probs=np.reshape(probs,probs.shape
[1])

ix=np.random.choice(range(vocab_si
ze),p=probs.ravel())

generated+=indices_char[ix]
    return generated
```

6. Output

6.1 Accuracy- epoch by epoch

```
Epoch 1/11
367332/367332 [==============================] - 320s 871us/step - loss: 2.9066
Epoch 2/11
367332/367332 [==============================] - 319s 870us/step - loss: 2.8427
Epoch 3/11
367332/367332 [==============================] - 320s 870us/step - loss: 2.8419
Epoch 4/11
367332/367332 [==============================] - 320s 871us/step - loss: 2.8428
Epoch 5/11
367332/367332 [==============================] - 320s 871us/step - loss: 2.8411
Epoch 6/11
367332/367332 [==============================] - 320s 872us/step - loss: 2.8410
Epoch 7/11
367332/367332 [==============================] - 319s 869us/step - loss: 3.0069
Epoch 8/11
367332/367332 [==============================] - 320s 870us/step - loss: 3.0297
Epoch 9/11
367332/367332 [==============================] - 320s 871us/step - loss: 3.0308
Epoch 10/11
367332/367332 [==============================] - 319s 869us/step - loss: 3.0303
Epoch 11/11
367332/367332 [==============================] - 319s 870us/step - loss: 3.0302
<keras.callbacks.History at 0x7ff2f6bb57b8>
```

6.2 Lyric Generation of drake songs

6.2.1 Dataset of Lyrics of Drake songs

```
[1] import numpy as np
    import pandas as pd
```

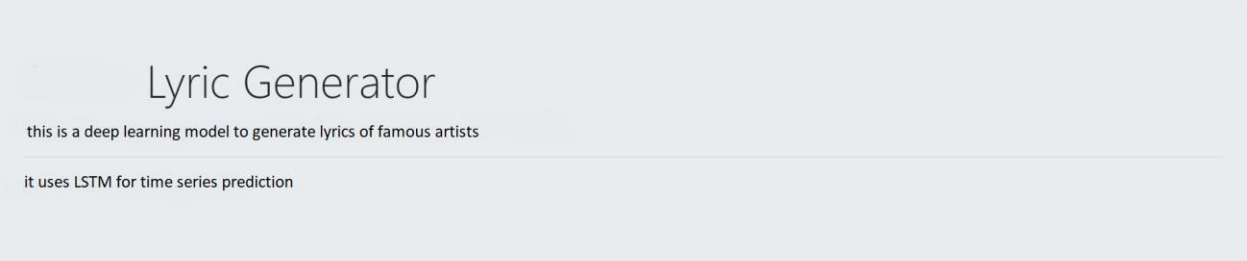
```
from google.colab import files
uploaded = files.upload()
```

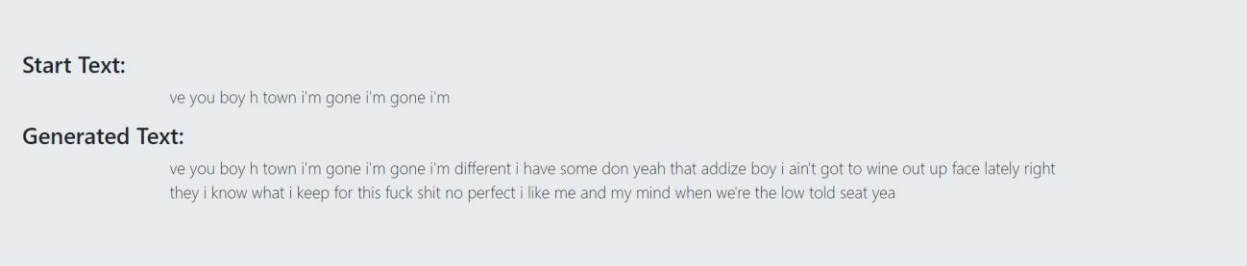
Choose Files drake-songs.csv

- **drake-songs.csv**(application/vnd.ms-excel) - 398640 bytes, last modified: 11/8/2019 - 100% done

```
Saving drake-songs.csv to drake-songs.csv
```

6.2.1 Lyrics Generated in the app

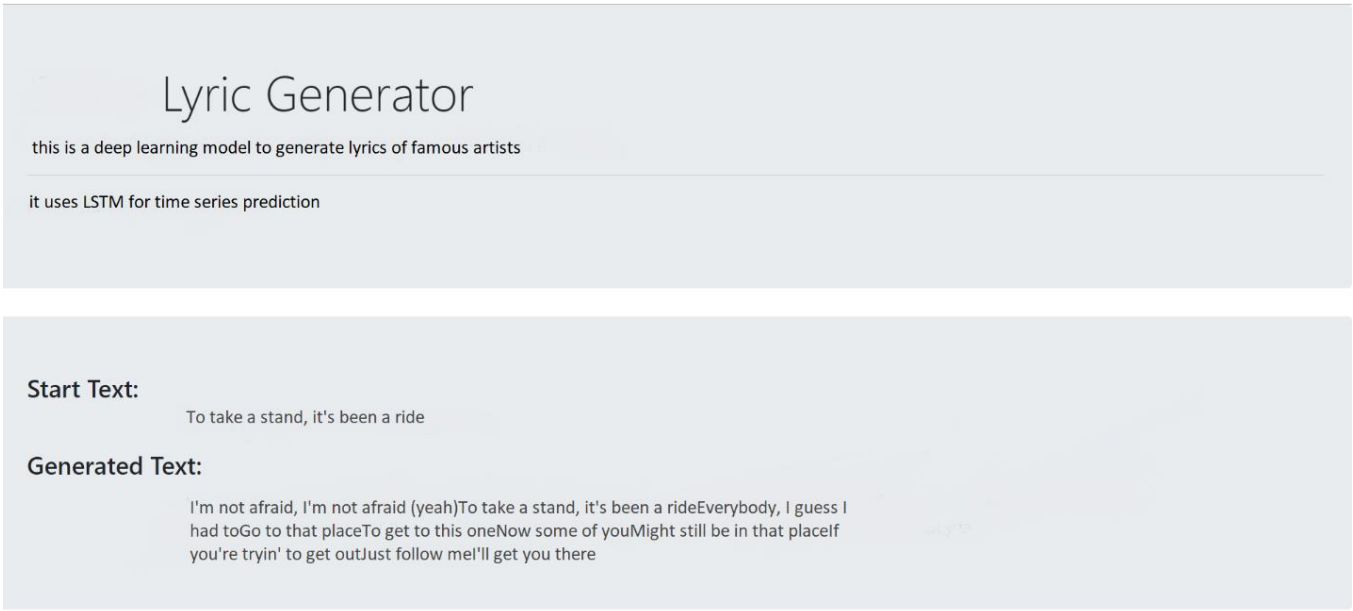




6.3 Lyric Generation of eminem songs

6.2.1 Dataset of Lyrics of eminem songs

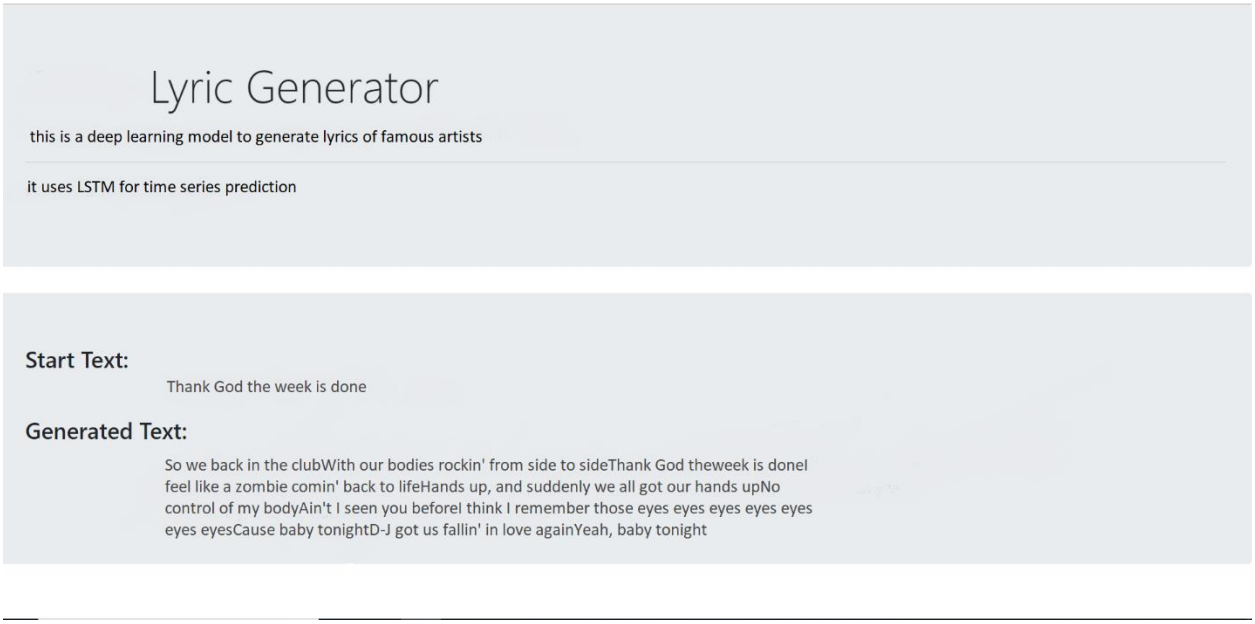
6.3.2 Lyrics Generated in the app.



6.4Lyric Generation of Usher songs

6.4.1 Dataset of Lyrics of Usher songs

6.4.2 Lyrics generated in the app



7 Conclusion:

This paper presents the Algorithm, code and Platform for the Lyric Generator which is able to generate the Lyrics of the song. This model is developed with 98% Accuracy and able to Generate the Lyrics of the songs Accurately. This Algorithm is also helpful in Industrial Applications like Text suggestions on keyboards, Schematic approach to Natural Language Processing, Auto fillers for ease, Automated email generation systems, Chatbots Etc…. In this way we developed a successful and useful application and created a platform which generates the Lyrics accurately by using LSTM's.

8. References

[1] Semantically Conditioned LSTM-based Natural Language Generation for Spoken Dialogue Systems

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, Steve Young

(Submitted on 7 Aug 2015 (v1), last revised 26 Aug 2015 (this version, v2))

[2] A Hierarchical Neural Autoencoder for Paragraphs and Documents

Jiwei Li, Minh-Thang Luong, Dan Jurafsky

(Submitted on 2 Jun 2015 (v1), last revised 6 Jun 2015 (this version, v2))

[3]https://www.researchgate.net/publication/49600679_Neural_networks_used_for_speech_recognition

Neural Networks used for Speech Recognition Wouter Gevaert, Georgi Tsenov, Valeri Mladenov, Senior Member, IEEE

[4] Rabiner L., Bing_Hwang J.,"Fundamentals of Speech Recognition", Prentice_Hall, 1993.

[5]Jurafsky, Daniel and Martin, James H, "Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition" (1st ed.). Prentice Hall, 1996

[6] Chollet, F.: Keras: Deep learning library for theano and tensorflow.

https://github.com/fchollet/keras (2015)

[7] Eck, D., Schmidhuber, J.: A first look at music composition using lstm recurrent

neural networks 103 (2002)

[8] https://arxiv.org/abs/1508.01745

Semantically Conditioned LSTM-based Natural Language Generation for Spoken Dialogue Systems

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, Steve Young

(Submitted on 7 Aug 2015 (v1), last revised 26 Aug 2015 (this version, v2))

[9] Thomas Hauk, Michael Buro and Jonathan Schaeffer, "*- Minimax Performance in Backgammon", Computer and Games Conference, Ramat-Gan 2004, pg 61-66 17. Russell Stuart J., Norvig Peter. "Artificial Intelligence. A modern approach.". Prentice Hall, 1995. ISBN 0-13-103805

[10] B. Huang, "Pruning Game Tree by Rollouts", 29th AAAI Conference on Artificial Intelligence, pp. 1165-1173, 2015.

[11] . S. Gal, “A discrete search game”, SIAM Journal of Applied Mathematics 27(4), pp. 641-648, 1974. 10. H. W. Kuhn, Classics in Game Theory, Princeton University Press, 1997.

[12]. Anurag Bhatt, Pratul Varshney, and Kalyanmoy Deb “In Search of No-loss Strategies for the Game of Tic-Tac-Toe using a Customized Genetic Algorithm” ,Pages 889-896, 2008, Proceedings of the 10th annual conference on Genetic and evolutionary computation, Atlanta, GA, USA.

[13]. Plamenka Borovska, Milena Lazarova, “Efficiency of parallel minimax algorithm for game tree search”, ACM InternationalConference Proceeding Series; Vol. 285 Proceedings of the 2007 international conference on Computer systems Bulgaria Article No. 14 Year of Publication: 2007 ISBN:978-954-9641-50-9.

[14] Tsung-Hsien Wen is qualified to endorse.

Semantically Conditioned LSTM-based Natural Language Generation for Spoken Dialogue Systems