Phish Setlist Generator

By Kenny Callaghan

Introduction: Problem

Phish is an American rock band formed in 1983 in Burlington, VT. Over the course of their 30+ years of prolific touring, the band has gained a reputation for their whimsical sense of humor, whacky onstage antics, and virtuistic instrumental improvisation. All of this has underscored the quintessential aspect of a Phish concert: every show is unique, and no two shows are quite the same. The easiest way to ensure this is to develop a unique setlist (ordered set) of songs for each of their 1700+ shows. When a show begins, no one can be sure what songs will be played - not even the members of the band themselves! What if there was a way to predict a setlist before-hand?

Solution

We would like to create an algorithm that can create Phish setlists from scratch - without any input from the band. We could do this by assembling a random list of songs, but we can do better. We can inform what song comes next in a setlist by looking at historical setlist data and extrapolating from there. We can start with a randomly-chosen song, either from a list of all previously-played songs, or from a smaller pool of all songs that have been used before as a set-opener. Then we can use a Natural Language Processing (NLP) machine learning algorithm to predict the next songs that can be played in the setlist - similar to the word prediction system that is used by Google and other services to autocomplete searches or messages. The model will look at an ordered list of songs of a chosen LENGTH, and use that to predict which songs are likely to come next. The model is a Long Short Term Memory (LSTM) network which is a type of Recurrent Neural Network (RNN). The model will be trained on 80% of existing Phish setlists, adjusting the weights of the feature parameters as it works through the training set. Then, the model will be tested against the remaining 20% of setlists to determine how accurate it's prediction is. Once we are able to predict a subsequent song given a sequence of songs, we can begin to generate setlists from scratch, generating predictions based off of the existing setlist.

In [127…

```
# This code will help for later

# Scientific and vector computation for python
import numpy as np

# Plotting library
from matplotlib import pyplot as plt

import tensorflow as tf
from nltk.tokenize import RegexpTokenizer
from keras.models import Sequential, load_model
from keras.layers import LSTM
from keras.layers.core import Dense, Activation
from keras.optimizers import RMSprop
import pickle
import heapq
```

Data

The dataset we are using is the entire record of every live performance from the band. This data was scraped from phish.net, a fan-run site that keeps detailed records of everything Phish-related. The data was then sanitized and input into a sqlite database with a schema in Third Normal Form. The code for this process can be found at this public github repository: https://github.com/kcallaghan1/Phish-DB. The final step, was to output the view of ("setInfo" [SET 1, SET 2, ENCORE], "songTitle") into a text document "setSongs.txt". We can then run the following cell to remove the setInfo from in front of a songTitle, and include it in the list of songs, as if it were a song itself. This is done to create context in a sublist of songs randomly selected from the dataset.

In [2]:

```
fr = open("setSongs.txt", "r")
fw = open("songList.txt", "w")
lines = fr.readlines()

setInfo = ""
for i in range(1, len(lines)):
    line = lines[i]
    first, second = line.split(",", 1)

    if("\"" not in first):
        first = "\"" + first + "\""
    if(setInfo != first):
        if(first == "\"SET 1\""):
            fw.write("\"END\"\n")
        setInfo = first
        if("\"" not in setInfo):
            setInfo = "\"" + first + "\""
        fw.write(setInfo + "\n")
    if("\"" not in second):
        second = "\"" + second[0: len(second)-1] + "\"\n"
    fw.write(second)

fr.close()
fw.close()

fr = open("songList.txt", "r")
songs = fr.readlines()
songs = songs[1:len(songs) - 1]
songs.append("\"More\"\n")
songs.append("\"END\"\n")
fr.close()

print(songs)
```


['"SET 1"\n', '"Long Cool Woman in a Black Dress"\n', '"Proud Mary"\n', '"In the Midnight Hour"\n', '"Squeeze Box"\n', '"Roadhouse Blues"\n', '"Happy Birthday to You"\n', '"SET 2"\n', '"Scarlet Begonias"\n', '"Fire on the Mountain"\n', '"END"\n', '"SET 1"\n', '"Makisupa Policeman"\n', '"In the Midnight Hour"\n', '"Wild Child"\n', '"Jam"\n', '"Bertha"\n', '"Can\'t You Hear Me Knocking"\n', '"St. Stephen Jam"\n', '"Can\'t You Hear Me Knocking"\n', '"Camel Walk"\n', '"Eyes of the World"\n', '"Whipping Post"\n', '"Drums"\n', '"Jam"\n', '"Wild Child"\n', '"Bertha"\n', '"Can\'t You Hear Me Knocking"\n', '"Camel Walk"\n', '"Jam"\n', '"In the Midnight Hour"\n', '"Scarlet Begonias"\n', '"Fire"\n', '"Fire on the Mountain"\n', '"Makisupa Policeman"\n', '"Slave to the Traffic Light"\n', '"Spanish Flea"\n', '"Don\'t Want You No More"\n', '"Cities"\n', '"Drums"\n', '"Skippy the Wondermouse"\n', '"Fluffhead"\n', '"ENCORE"\n', '"Eyes of the World"\n', '"END"\n', '"SET 1"\n', '"Slave to the Traffic Light"\n', '"Mike\'s Song"\n', '"Dave\'s Energy Guide"\n', '"You Enjoy Myself"\n', '"Alumni Blues"\n', '"Letter to Jimmy Page"\n', '"Alumni Blues"\n', '"Prep School Hippie"\n', '"Run Like an Antelope"\n', '"Anarchy"\n', '"Camel Walk"\n', '"Fire Up the Ganja"\n', '"Skippy the Wondermouse"\n', '"In the Midnight Hour"\n', '"Sneakin\' Sally Through the Alley"\n', '"Alumni Blues"\n', '"Skippy the Wondermouse"\n', '"Bertha"\n', '"Jam"\n', '"Mike\'s Song"\n', '"I Am Hydrogen"\n', '"Don\'t Want You No More"\n', '"Cities"\n', '"Wild Child"\n', '"Slave to the Traffic Light"\n', '"Jam"\n', '"Alumni Blues"\n', '"Letter to Jimmy Page"\n', '"Alumni Blues"\n', '"Mike\'s Song"\n', '"Dave\'s Energy Guide"\n', '"Big Leg Emma"\n', '"Revolution"\n', '"Anarchy"\n', '"I Wish"\n', '"Camel Walk"\n', '"Run Like an Antelope"\n', '"SET 2"\n', '"McGrupp and the Watchful Hosemasters"\n', '"Makisupa Policeman"\n', '"SET 3"\n', '"Quinn the Eskimo"\n', '"END"\n', '"SET 1"\n', '"Slave to the Traffic Light"\n', '"Mike\'s Song"\n', '"Dave\'s Energy Guide"\n', '"Big Leg Emma"\n', '"SET 2"\n', '"Alumni Blues"\n', '"Wild Child"\n', '"Can\'t You Hear Me Knocking"\n', '"Jam"\n', '"Cities"\n', '"Bring It On Home"\n', '"SET 3"\n', '"Scarlet Begonias"\n', '"Eyes of the World"\n', '"Whipping Post"\n', '"McGrupp and the Watchful Hosemasters"\n', '"Makisupa Policeman"\n', '"Run Like an Antelope"\n', '"The Other One"\n', '"ENCORE"\n', '"Anarchy"\n', '"END"\n', '"SET 1"\n', '"Big Leg Emma"\n', '"Dog Log"\n', '"Prep School Hippie"\n', '"Camel Walk"\n', '"Alumni Blues"\n', '"Run Like an Antelope"\n', '"Dave\'s Energy Guide"\n', '"McGrupp and the Watchful Hosemasters"\n', '"Wild Child"\n', '"Back Porch Boogie Blues"\n', '"Camel Walk"\n', '"Possum"\n', '"Fire"\n', '"Slave to the Traffic Light"\n', '"In the Midnight Hour"\n', '"SET 2"\n', '"Melt the Guns"\n', '"T.V. Theme"\n', '"Sneakin\' Sally Through the Alley"\n', '"Prep School Hippie"\n', '"SET 3"\n', '"Mike\'s Song"\n', '"Dave\'s Energy Guide"\n', '"McGrupp and the Watchful Hosemasters"\n', '"Cities"\n', '"Anarchy"\n', '"Revolution"\n', '"Anarchy"\n', '"Revolution"\n', '"Makisupa Policeman"\n', '"END"\n', '"SET 1"\n', '"T.V. Theme"\n', '"Letter to Jimmy Page"\n', '"Alumni Blues"\n', '"Mike\'s Song"\n', '"Dave\'s Energy Guide"\n', '"Revolution"\n', '"Anarchy"\n', '"Camel Walk"\n', '"Run Like an Antelope"\n', '"McGrupp and the Watchful Hosemasters"\n', '"Harry Hood"\n', '"Dog Log"\n', '"Possum"\n', '"Slave to the Traffic Light"\n', '"Sneakin\' Sally Through the Alley"\n', '"I Wish"\n', '"Revival"\n', '"Alumni Blues"\n', '"Letter to Jimmy Page"\n', '"Alumni Blues"\n', '"Prep School Hippie"\n', '"Skippy the Wondermouse"\n', '"Slave to the Traffic Light"\n', '"Hurricane"\n', '"Makisupa Policeman"\n', '"Piggies"\n', '"Makisupa Policeman"\n', '"Drums"\n', '"Alumni Blues"\n', '"Dear Mrs. Reagan"\n', '"Wild Thing"\n', '"SET 2"\n', '"Mike\'s Song"\n', '"Whipping Post Jam"\n', '"Run Like an Antelope"\n', '"Dave\'s Energy Guide"\n', '"END"\n', '"SET 1"\n', '"Slave to the Traffic Light"\n', '"Mike\'s Song"\n', '"Dave\'s Energy Guide"\n', '"You Enjoy Myself"\n', '"Alumni Blues"\n', '"Letter to Jimmy Page"\n', '"Alumni Blues"\n', '"Prep School Hippie"\n', '"Run Like an Antelope"\n', '"SET 2"\n', '"Tonight"\n', '"The Pendulum"\n', '"Babylon is Burning"\n', '"Dec 1661"\n', '"END"\n', '"SET 1"\n', '"Quinn the Eskimo"\n', '"Have Mercy"\n', '"Harry Hood"\n', '"The Pendulum"\n', '"Dave\'s Energy Guide"\n', '"Icculus"\n', '"You Enjoy Myself"\n', '"SET 2"\n', '"Help on the Way"\n', '"Slipknot!"\n', '"AC/DC Bag"\n', '"McGrupp and the Watchful Hosemasters"\n', '"Alumni Blues"\n', '"Letter to Jimmy Page"\n', '"Alumni Blues"\n', '"Dear Mrs. Reagan"\n', '"ENCORE"\n', '"Not Fade Away"\n', '"END"\n', '"SET 1"\n', '"AC/DC Bag"\n', '"Dear Mrs. Reagan"\n', '"Prep School Hippie"\n', '"Quinn the Eskimo"\n', '"Slave to the Traffic Light"\n', ...

Enumerating songs

Next, we will need to create a sorted array of unique songs that we can index and use in our model.

```
unique_songs = sorted(list(set(songs)))
print(unique_songs)
```

Splitting the Data

The code will run through the entire dataset, sending 80% of current vertices to the training set, and the remaining 20% to the test set. Remember that the vertices are stored in the variable `vertices`, and the labels are stored in the variable `labels`. Also, the split must occur between entries from the same user, so that each dataset (training and test) fully represent entire tracks from the start of the user that song.

To accomplish, we will iterate through the entire vertices chronologically, counting up for each existing song. We use `train` before the first split, and `test` after the split.

```
split_index = 0
for i in range(len(vertices)):
    if vertices[i][0] != vertices[i-1][0]:
        if split_index == 0:
            split_index = int(0.8 * (i-1))
        else:
            break

train_vertices = vertices[:split_index]
test_vertices = vertices[split_index:]
```

```
train_labels = labels[:split_index]
test_labels = labels[split_index:]
```

s"\n', '"Harry Hood"\n', '"Shine a Light"\n', '"Julius"\n', '"ENCORE"\n', '"Love Is What We Are"\n', '"Golgi Apparatus"\n', '"END"\n', '"SET 1"\n', '"Stealing Time From the Faulty Plan"\n', '"The Moma Dance"\n', '"The Wedge"\n', '"Halfway to the Moon"\n', '"Ya Mar"\n', '"Martian Monster"\n', '"Party Time"\n', '"Wingsuit"\n', '"Bouncing Around the Room"\n', '"More"\n', '"SET 2"\n', '"Corona"\n', '"Simple"\n', '"Winterqueen"\n', '"Light"\n', '"Scents and Subtle Sounds"\n', '"Cities"\n', '"Slave to the Traffic Light"\n', '"ENCORE"\n', '"Loving Cup"\n', '"END"\n', '"SET 1"\n', '"Chocolate Rain"\n', '"Ass Handed"\n', '"Free"\n', '"Weigh"\n', '"Undermind"\n', '"The Oh Kee Pa Ceremony"\n', '"The Dogs"\n', '"Destiny Unbound"\n', '"Divided Sky"\n', '"Things People Do"\n', '"Sand"\n', '"SET 2"\n', '"Have Mercy"\n', '"Chalk Dust Torture"\n', '"You Sexy Thing"\n', '"Mercury"\n', '"You Sexy Thing"\n', '"Backwards Down the Number Line"\n', '"Rock and Roll"\n', '"ENCORE"\n', '"Fee"\n', '"Space Oddity"\n', '"END"\n', '"SET 1"\n', '"Llama"\n', '"Wilson"\n', '"Stealing Time From the Faulty Plan"\n', '"Ya Mar"\n', '"Tela"\n', '"The Birds"\n', '"The Line"\n', '"Water in the Sky"\n', '"Vultures"\n', '"Train Song"\n', '"Horn"\n', '"I Am the Walrus"\n', '"SET 2"\n', '"Blaze On"\n', '"Twenty Years Later"\n', '"Alumni Blues"\n', '"Letter to Jimmy Page"\n', '"Alumni Blues"\n', '"Meatstick"\n', '"Dirt"\n', '"Harry Hood"\n', '"ENCORE"\n', '"Cinnamon Girl"\n', '"END"\n', '"SET 1"\n', '"Buffalo Bill"\n', '"The Moma Dance"\n', '"Birds of a Feather"\n', '"Sugar Shack"\n', '"Most Events Aren\'t Planned"\n', '"Back on the Train"\n', '"Leaves"\n', '"The Wedge"\n', '"46 Days"\n', '"Bathtub Gin"\n', '"SET 2"\n', '"Down with Disease"\n', '"Rise/Come Together"\n', '"Piper"\n', '"Meatstick"\n', '"Also Sprach Zarathustra"\n', '"Possum"\n', '"Suzy Greenberg"\n', '"ENCORE"\n', '"Waste"\n', '"First Tube"\n', '"END"\n', '"SET 1"\n', '"AC/DC Bag"\n', '"Wolfman\'s Brother"\n', '"Roggae"\n', '"Tube"\n', '"Bouncing Around the Room"\n', '"Back on the Train"\n', '"Your Pet Cat"\n', '"Waking Up Dead"\n', '"Theme From the Bottom"\n', '"SET 2"\n', '"Wilson"\n', '"No Men In No Man\'s Land"\n', '"Twist"\n', '"Everything\'s Right"\n', '"Also Sprach Zarathustra"\n', '"Harry Hood"\n', '"ENCORE"\n', '"The Wedge"\n', '"Slave to the Traffic Light"\n', '"END"\n', '"SET 1"\n', '"46 Days"\n', '"McGrupp and the Watchful Hosemasters"\n', '"Pigtail"\n', '"Cities"\n', '"Nellie Kane"\n', '"Gumbo"\n', '"Guyute"\n', '"Axilla"\n', '"The Dogs"\n', '"Dirt"\n', '"David Bowie"\n', '"SET 2"\n', '"The Moma Dance"\n', '"A Song I Heard the Ocean Sing"\n', '"Mercury"\n', '"Carini"\n', '"Maze"\n', '"Boogie On Reggae Woman"\n', '"Harry Hood"\n', '"ENCORE"\n', '"The Squirming Coil"\n', '"END"\n', '"SET 1"\n', '"Roggae"\n', '"Tube"\n', '"NICU"\n', '"Runaway Jim"\n', '"The Horse"\n', '"Silent in the Morning"\n', '"Keeping It Real"\n', '"Driver"\n', '"Saw It Again"\n', '"Ocelot"\n', '"Waking Up Dead"\n', '"Backwards Down the Number Line"\n', '"More"\n', '"SET 2"\n', '"Set Your Soul Free"\n', '"Twist"\n', '"Makisupa Policeman"\n', '"Scents and Subtle Sounds"\n', '"What\'s the Use?"\n', '"The Wedge"\n', '"Possum"\n', '"ENCORE"\n', '"The Lizards"\n', '"END"\n', '"SET 1"\n', '"Stealing Time From the Faulty Plan"\n', '"Funky Bitch"\n', '"Ocelot"\n', '"The Wedge"\n', '"Wombat"\n', '"Guelah Papyrus"\n', '"Birds of a Feather"\n', '"Saw It Again"\n', '"Timber (Jerry The Mule)"\n', '"Limb By Limb"\n', '"Farmhouse"\n', '"More"\n', '"SET 2"\n', '"Meatstick"\n', '"Drowned"\n', '"NICU"\n', '"Thread"\n', '"Runaway Jim"\n', '"Run Like an Antelope"\n', '"Runaway Jim"\n', '"Run Like an Antelope"\n', '"Makisupa Policeman"\n', '"Run Like an Antelope"\n', '"You Enjoy Myself"\n', '"ENCORE"\n', '"A Day in the Life"\n', '"END"\n', '"SET 1"\n', '"Blaze On"\n', '"Party Time"\n', '"Breath and Burning"\n', '"Sugar Shack"\n', '"Home"\n', '"Joy"\n', '"Stash"\n', '"46 Days"\n', '"SET 2"\n', '"Mercury"\n', '"Ghost"\n', '"Fuego"\n', '"Slave to the Traffic Light"\n', '"ENCORE"\n', '"Twenty Years Later"\n', '"Martian Monster"\n', '"Rocky Top"\n', '"Golgi Apparatus"\n', '"END"\n', '"SET 1"\n', '"Stealing Time From the Faulty Plan"\n', '"Skin It Back"\n', '"Brian and Robert"\n', '"Timber (Jerry The Mule)"\n', '"Simple"\n', '"Mexican Cousin"\n', '"Camel Walk"\n', '"Back on the Train"\n', '"Saw It Again"\n', '"SET 2"\n', '"Waves"\n', '"Rise/Come Together"\n', '"Light"\n', '"The Line"\n', '"Wingsuit"\n', '"Your Pet Cat"\n', '"What\'s the Use?"\n', '"Possum"\n', '"ENCORE"\n', '"More"\n', '"ENCORE 2"\n', '"You Enjoy Myself"\n', '"END"\n', '"SET 1"\n', '"Tweezer Reprise"\n', '"Party Time"\n', '"Free"\n', '"The Old Home Place"\n', '"Lawn Boy"\n', '"Cool It Down"\n', '"Theme From the Bottom"\n', '"Funky Bitch"\n', '"Kill Devil Falls"\n', '"Wolfman\'s Brother"\n', '"Character Zero"\n', '"SET 2"\n', '"Mike\'s Song"\n', '"Ghost"\n', '"Everything\'s Right"\n', '"Set Your Soul Free"\n', '"When the Circus Comes"\n', '"Weekapaug Groove"\n', '"ENCORE"\n', '"Loving Cup"\n', '"END"\n', '"SET 1"\n', '"We Are Come to Outlive Our Brains"\n', '"Martian Monster"\n', '"Axilla"\n', '"Free"\n', '"The Wedge"\n', '"Meat"\n', '"Ghost"\n', '"Sparkle"\n', '"If I Could"\n', '"Maze"\n', '"Walls of the Cave"\n', '"SET 2"\n', '"Set Your Soul Free"\n', '"Swept Away"\n', '"Steep"\n', '"The Final Hurrah"\n', '"Fuego"\n', '"Shade"\n', '"Bathtub Gin"\n', '"Possum"\n', '"ENCORE"\n', '"Bouncing Around the Room"\n', '"Slave to the Traffic Light"\n', '"END"\n', '"SET 1"\n', '"Buried Alive"\n', '"Blaze On"\n', '"Turtle in the Clouds"\n', '"The Sloth"\n', '"46 Days"\n', '"Cities"\n', '"Corinna"\n', '"Ya Mar"\n', '"Wolfman\'s Brother"\n', '"SET 2"\n', '"Carini"\n', '"Tweezer"\n', '"Death Don\'t Hurt Very Long"\n', '"Tweezer"\n', '"No Quarter"\n', '"Also Sprach Zarathustra"\n', '"First Tube"\n', '"ENCORE"\n', '"Shine a Light"\n', '"Tweezer Reprise"\n', '"END"\n', '"SET 1"\n', '"Set Your Soul Free"\n', '"Blaze On"\n', '"Death Don\'t Hurt Very Long"\n', '"Reba"\n', '"Free"\n', '"Sand"\n', '"Wolfman\'s Brother"\n', '"Cavern"\n', '"SET 2"\n', '"Mike\'s Song"\n', '"Fluffhead"\n', '"Twist"\n', '"Weekapaug Groove"\n', '"No Men In No Man\'s Land"\n', '"Weekapaug Groove"\n', '"Twist"\n', '"Fuego"\n', '"Ghost"\n', '"Bathtub Gin"\n', '"ENCORE"\n', '"Wilson"\n', '"First Tube"\n', '"END"\n', '"SET 1"\n', '"AC/DC Bag"\n', '"555"\n', '"Ocelot"\n', '"Sample in a Jar"\n', '"Stash"\n', '"The Wedge"\n', '"Frost"\n', '"Halley\'s Comet"\n', '"Ruby Waves"\n', '"Lawn Boy"\n', '"You Enjoy Myself"\n', '"SET 2"\n', '"Plasma"\n', '"The Final Hurrah"\n', '"Wingsuit"\n', '"Golden Age"\n', '"I Always Wanted It This Way"\n', '"Prince Caspian"\n', '"If I Could"\n', '"46 Days"\n', '"ENCORE"\n', '"Drift While You\'re Sleeping"\n', '"END"\n', '"SET 1"\n', '"The Curtain With"\n', '"Fast Enough for You"\n', '"Buried Alive"\n', '"Camel Walk"\n', '"Reba"\n', '"Sample in a Jar"\n', '"Pebbles and Marbles"\n', '"Tela"\n', '"The Mango Song"\n', '"Driver"\n', '"David Bowie"\n', '"SET 2"\n', '"Mr. Completely"\n', '"Twenty Years Later"\n', '"Big Black Furry Creature from Mars"\n', '"Tweezer"\n', '"Shade"\n', '"Most Events Aren\'t Planned"\n', '"Makisupa Policeman"\n', '"Chalk Dust Torture"\n', '"Suzy Greenberg"\n', '"ENCORE"\n', '"Punch You in the Eye"\n', '"What\'s the Use?"\n', '"Julius"\n', '"END"\n', '"SET 1"\n', '"Cathy\'s Clown"\n', '"Tweezer Reprise"\n', '"Carini"\n', '"AC/DC Bag"\n', '"The Moma Dance"\n', '"Theme From the Bottom"\n', '"Meat"\n', '"Home"\n', '"Bathtub Gin"\n', '"Walls of the Cave"\n', '"SET 2"\n', '"Cool Amber and Mercury"\n', '"Down with Disease"\n', '"Scents and Subtle Sounds"\n', '"Twist"\n', '"Wilson"\n', '"Scent of a Mule"\n', '"Fuck Your Face"\n', '"Halley\'s Comet"\n', '"Harry Hood"\n', '"ENCORE"\n', '"Fee"\n', '"A Life Beyond The Dream"\n', '"First Tube"\n', '"END"\n', '"SET 1"\n', '"Can\'t Always Listen"\n', '"Free"\n', '"No Men In No Man\'s Land"\n', '"555"\n', '"Back on the Train"\n', '"Rift"\n', '"Poor Heart"\n', '"Undermind"\n', '"Train Song"\n', '"Wingsuit"\n', '"Blaze On"\n', '"SET 2"\n', '"Everything\'s Right"\n', '"Mercury"\n', '"Shade"\n', '"Light"\n', '"Party Time"\n', '"Sand"\n', '"Harry Hood"\n', '"ENCORE"\n', '"Say It To Me S.A.N.T.O.S."\n', '"END"\n', '"SET 1"\n', '"The Moma Dance"\n', '"The Final Hurrah"\n', '"Gumbo"\n', '"Access Me"\n', '"Funky Bitch"\n', '"Ghost"\n', '"Tube"\n', '"Mountains in the Mist"\n', '"Drift While You\'re Sleeping"\n', '"SET 2"\n', '"Mike\'s Song"\n', '"I Am Hydrogen"\n', '"Weekapaug Groove"\n', '"46 Days"\n', '"Set Your Soul Free"\n', '"Down with Disease"\n', '"NICU"\n', '"Bug"\n', '"The Wedge"\n', '"Slave to the Traffic Light"\n', '"ENCORE"\n', '"Brian and Robert"\n', '"Character Zero"\n', '"END"\n', '"SET 1"\n', '"Sigma Oasis"\n', '"Buried Alive"\n', '"Guelah Papyrus"\n', '"My Sweet One"\n', '"Everything is Hollow"\n', '"The Curtain"\n', '"Mound"\n', '"Brian and Robert"\n', '"Crazy Sometimes"\n', '"Frost"\n', '"Breath and Burning"\n', '"Alumni Blues"\n', '"Letter to Jimmy Page"\n', '"Alumni Blues"\n', '"It\'s Ice"\n', '"Walls of the Cave"\n', '"SET 2"\n', '"After Midnight"\n', '"Stealing Time From the Faulty Plan"\n', '"Energy"\n', '"Soul Planet"\n', '"Wingsuit"\n', '"I Always Wanted It This Way"\n', '"Petrichor"\n', '"Boogie On Reggae Woman"\n', '"Rise/Come Together"\n', '"ENCORE"\n', '"Pebbles and Marbles"\n', '"Beneath a Sea of Stars Part 1"\n', '"Izabella"\n', '"Chalk Dust Torture Reprise"\n', '"END"\n', '"SET 1"\n', '"Evening Song"\n', '"No Men In No Man\'s Land"\n', '"Down with Disease"\n', '"20-20 Vision"\n', '"Ghost"\n', '"Gumbo"\n', '"Rift"\n', '"Weigh"\n', '"Dog Faced Boy"\n', '"Twenty Years Later"\n', '"Tube"\n', '"Say It To Me S.A.N.T.O.S."\n', '"SET 2"\n', '"Everything\'s Right"\n', '"Beneath a Sea of Stars Part 1"\n', '"Set Your Soul Free"\n', '"Gotta Jibboo"\n', '"Free"\n', '"Piper"\n', '"Drowned"\n', '"Ass Handed"\n', '"Chalk Dust Torture Reprise"\n', '"Character Zero"\n', '"ENCORE"\n', '"A Life Beyond The Dream"\n', '"Rocky Top"\n', '"END"\n']

Model Parameters

Here is a list of parameters that can be useful to modify that will have impact on the speed and accuracy of the model.

In [90]:
```
# The length of the song sequence that the model will look at to predict the next song.
SEQ_LENGTH = 5

# Number of nodes in the LSTM layer
LSTM_NODES = 128

# Number of epochs - how many times the layers are passed through (forward + backwards = 1 epoch)
NUM_EPOCHS = 2

# Learning Rate
LR = 0.01
```

Features

We create two features X, containing a sequence of given length songs, and Y, the subsequent song, or target value.

In [91]:
```
prev_songs = []
next_songs = []
for i in range(len(training_set) - SEQ_LENGTH):
    prev_songs.append(training_set[i:i + SEQ_LENGTH])
    next_songs.append(training_set[i + SEQ_LENGTH])

X = np.zeros((len(prev_songs), SEQ_LENGTH, len(unique_songs)), dtype = bool)
Y = np.zeros((len(next_songs), len(unique_songs)), dtype = bool)

for i, each_songs in enumerate(prev_songs):
    for j, each_song in enumerate(each_songs):
        X[i, j, unique_song_index[each_song]] = 1
    Y[i, unique_song_index[next_songs[i]]] = 1
```

Model Creation

Our model is a single-layer LSTM model with a number of nodes that can be specified above, a dense-layer (layer of fully-connected nodes), and an activation layer using a softmax (generalized sigmoid) function.

In [92]:
```
model = Sequential()
model.add(LSTM(LSTM_NODES, input_shape=(SEQ_LENGTH, len(unique_songs))))
model.add(Dense(len(unique_songs)))
model.add(Activation("softmax"))
```

Training the Model

Next, we train the model using an RMSprop optimizer, which is similar the gradient descent algorithm. Our learning rate is set by the LR parameter defined above.

In [93]:
```
optimizer = RMSprop(LR)
model.compile(loss="categorical_crossentropy", optimizer=optimizer, metrics=["accuracy"])
history = model.fit(X, Y, validation_split=0.05, batch_size=128, epochs = NUM_EPOCHS, shuffle = True).history
```

```
Epoch 1/2
246/246 [==============================] - 11s 33ms/step - loss: 5.1117 - accuracy: 0.1126 - val_loss: 5.3407 - val_accuracy: 0.1598
Epoch 2/2
246/246 [==============================] - 7s 29ms/step - loss: 4.0841 - accuracy: 0.2139 - val_loss: 5.3808 - val_accuracy: 0.1659
```

Lastly, we can save the model and its history (containing an evaluation of the model's loss and accuracy) for future retreival using the following lines of code:

In [94]:
```
model.save("keras_next_song_model.h5")
pickle.dump(history, open("history.p", "wb"))

model = load_model("keras_next_song_model.h5")
history = pickle.load(open("history.p", "rb"))
```

Using the Model to Generate Predicitons

Now that we have a trained model, we can use it to generate song predictions based on a subsequence of the test set. First, we use the prepare_input function to mark the indices corresponding to the songs contained in the sequence. Then, the sample function will select the indices with the highest probability of being chosen to be used for the prediction. Finally, the predict_completions function will map the predicted indices to a song and return the predictions.

In [144…
```
def prepare_input(text, seq_length = SEQ_LENGTH):
    x = np.zeros((1, seq_length, len(unique_songs)))
    for t, song in enumerate(text):
        x[0, t, unique_song_index[song]] = 1
```

```python
    return x


def sample(preds, top_n = 3):
    preds = np.asarray(preds).astype("float64")
    preds = np.log(preds)
    exp_preds = np.exp(preds)
    preds = exp_preds / np.sum(exp_preds)

    return heapq.nlargest(top_n, range(len(preds)), preds.take)


def predict_completions(text, n=3):
    x = prepare_input(text, n)
    preds = model.predict(x, verbose=0)[0]
    next_indices = sample(preds, n)
    return [unique_songs[idx] for idx in next_indices]
```

We can use this code to randomly predict songs in the test set.

In [111…

```python
rand = np.random.randint(len(test_set))
q = test_set[rand:rand+SEQ_LENGTH]
seq = q[0:len(q) - 1 ]
print("Sequence: ", seq)
print("Correct song: ", q[SEQ_LENGTH - 1])
print("Next possible songs: ", predict_completions(seq, SEQ_LENGTH))
```

```
Sequence:  ['"Mike\'s Song"\n', '"The Horse"\n', '"Silent in the Morning"\n', '"Punch You in the Eye"\n']
Correct song:  "McGrupp and the Watchful Hosemasters"

Next possible songs:  ['"Weekapaug Groove"\n', '"ENCORE"\n', '"SET 2"\n', '"Hold Your Head Up"\n', '"SET 3"\n']
```

Testing The Effect Various Parameters

Here, the code from above has been copied down to conveniently test the effects that changing various parameters has on the speed and accuracy of the model.

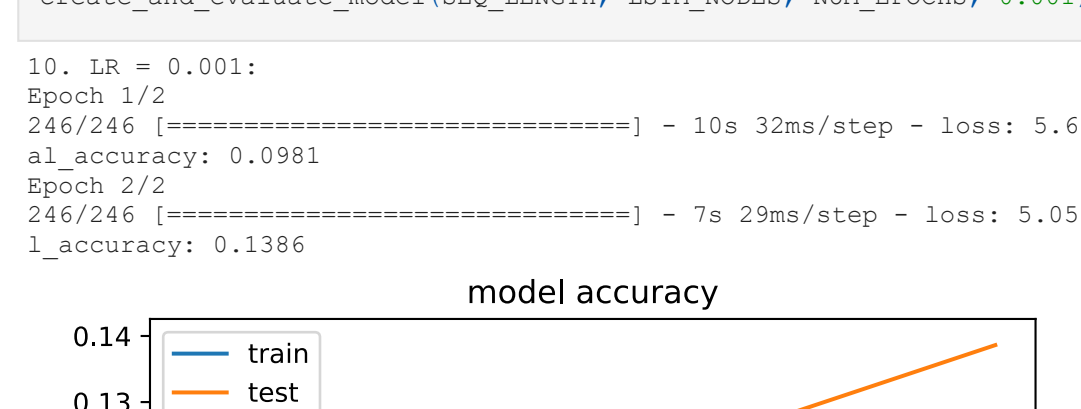
In [162…

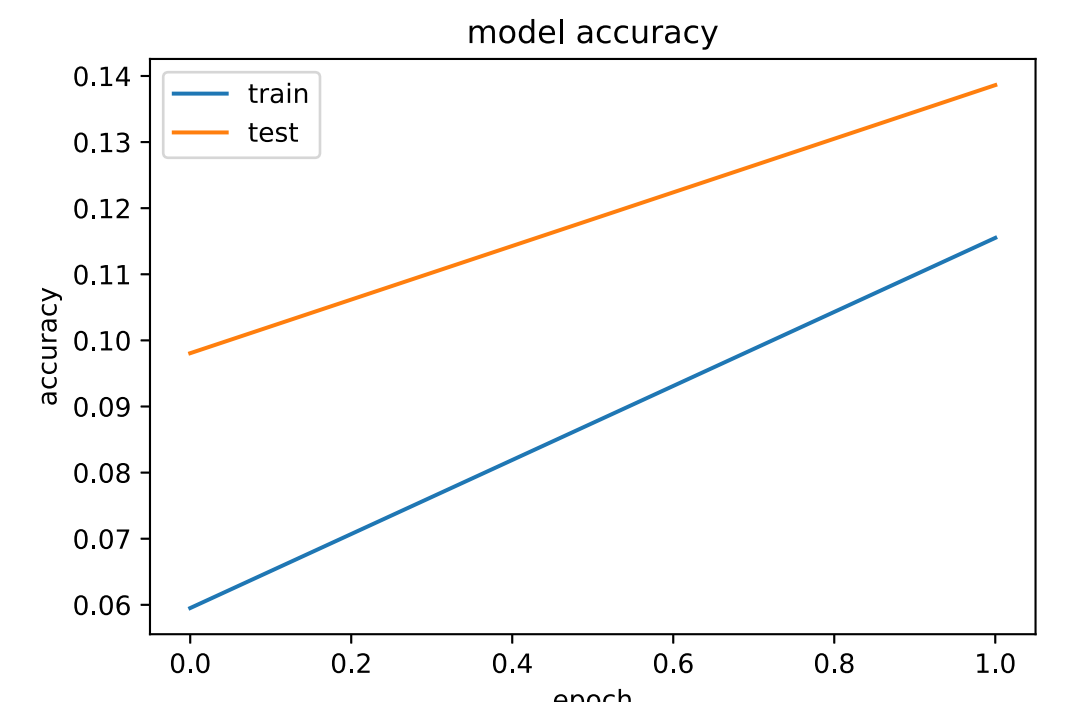
```python
def create_and_evaluate_model(seq_length, lstm_nodes, num_epochs, lr):
    prev_songs = []
    next_songs = []
    for i in range(len(training_set) - seq_length):
        prev_songs.append(training_set[i:i + seq_length])
        next_songs.append(training_set[i + seq_length])

    X = np.zeros((len(prev_songs), seq_length, len(unique_songs)), dtype = bool)
    Y = np.zeros((len(next_songs), len(unique_songs)), dtype = bool)

    for i, each_songs in enumerate(prev_songs):
        for j, each_song in enumerate(each_songs):
            X[i, j, unique_song_index[each_song]] = 1
        Y[i, unique_song_index[next_songs[i]]] = 1

    model = Sequential()
    model.add(LSTM(lstm_nodes, input_shape=(seq_length, len(unique_songs))))
    model.add(Dense(len(unique_songs)))
    model.add(Activation("softmax"))


    optimizer = RMSprop(lr)
    model.compile(loss="categorical_crossentropy", optimizer=optimizer, metrics=["accuracy"])
    history = model.fit(X, Y, validation_split=0.05, batch_size=128, epochs = num_epochs, shuffle = True).history

    # summarize history for accuracy
    plt.plot(history['accuracy'])
    plt.plot(history['val_accuracy'])
    plt.title('model accuracy')
    plt.ylabel('accuracy')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'], loc='upper left')
    plt.show()

    # summarize history for loss
    plt.plot(history['loss'])
    plt.plot(history['val_loss'])
    plt.title('model loss')
    plt.ylabel('loss')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'], loc='upper left')
    plt.show()

    correct = 0
    for i in range(seq_length,len(test_set) - seq_length):
        q = test_set[i:i+seq_length + 1]
        seq = q[0:len(q)-1]
        predicted_songs = predict_completions(seq, seq_length)
        if(q[len(q)-1] in predicted_songs):
            correct += 1
    print("Guessed", correct, "correct out of", len(test_set) - seq_length)
```

In [161…

```python
# The length of the song sequence that the model will look at to predict the next song.
SEQ_LENGTH = 5

# Number of nodes in the LSTM layer
LSTM_NODES = 128

# Number of epochs - how many times the layers are passed through (forward + backwards = 1 epoch)
NUM_EPOCHS = 2

# Learning Rate
LR = 0.01

# Testing various values:
print("1. Default values:")
create_and_evaluate_model(SEQ_LENGTH, LSTM_NODES, NUM_EPOCHS, LR)
print("2. SEQ_LENGTH = 10:")
create_and_evaluate_model(10, LSTM_NODES, NUM_EPOCHS, LR)
print("3. SEQ_LENGTH = 20:")
create_and_evaluate_model(20, LSTM_NODES, NUM_EPOCHS, LR)
print("4. LSTM_NODES = 50:")
create_and_evaluate_model(SEQ_LENGTH, 50, NUM_EPOCHS, LR)
print("5. LSTM_NODES = 150:")
create_and_evaluate_model(SEQ_LENGTH, 50, NUM_EPOCHS, LR)
print("6. NUM_EPOCHS = 5:")
create_and_evaluate_model(SEQ_LENGTH, LSTM_NODES, 5, LR)
print("7. NUM_EPOCHS = 10:")
create_and_evaluate_model(SEQ_LENGTH, LSTM_NODES, 10, LR)
print("8. NUM_EPOCHS = 20:")
create_and_evaluate_model(SEQ_LENGTH, LSTM_NODES, 20, LR)
print("9. LR = 0.1:")
create_and_evaluate_model(SEQ_LENGTH, LSTM_NODES, NUM_EPOCHS, 0.1)
print("10. LR = 0.001:")
create_and_evaluate_model(SEQ_LENGTH, LSTM_NODES, NUM_EPOCHS, 0.001)
```

```
10. LR = 0.001:
Epoch 1/2
246/246 [==============================] - 10s 32ms/step - loss: 5.6307 - accuracy: 0.0489 - val_loss: 6.0575 - v
al_accuracy: 0.0981
Epoch 2/2
246/246 [==============================] - 7s 29ms/step - loss: 5.0534 - accuracy: 0.1043 - val_loss: 5.9492 - va
l_accuracy: 0.1386
```

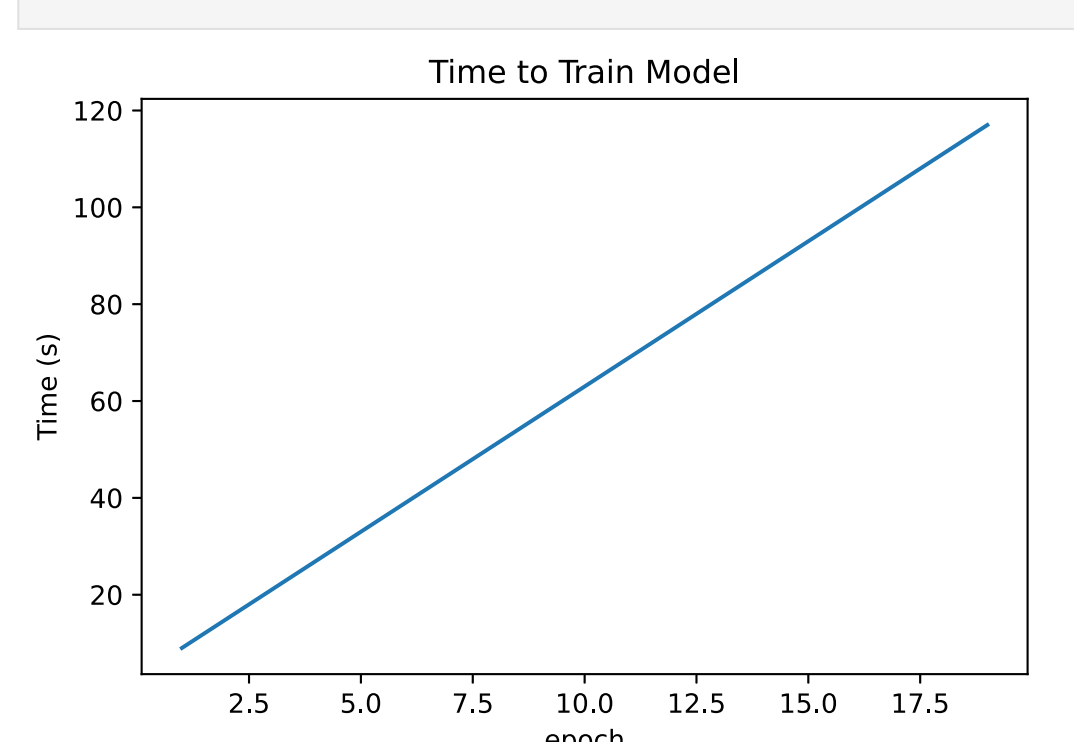





```
Guessed 18 correct out of 8217
```

In [181…

```python
epoch_num = [1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
epoch_time = [9, 21, 33, 45, 57, 69, 81, 93, 105, 117]
plt.plot(epoch_num, epoch_time)
plt.title('Time to Train Model')
plt.ylabel('Time (s)')
plt.xlabel('epoch')
plt.show()
```



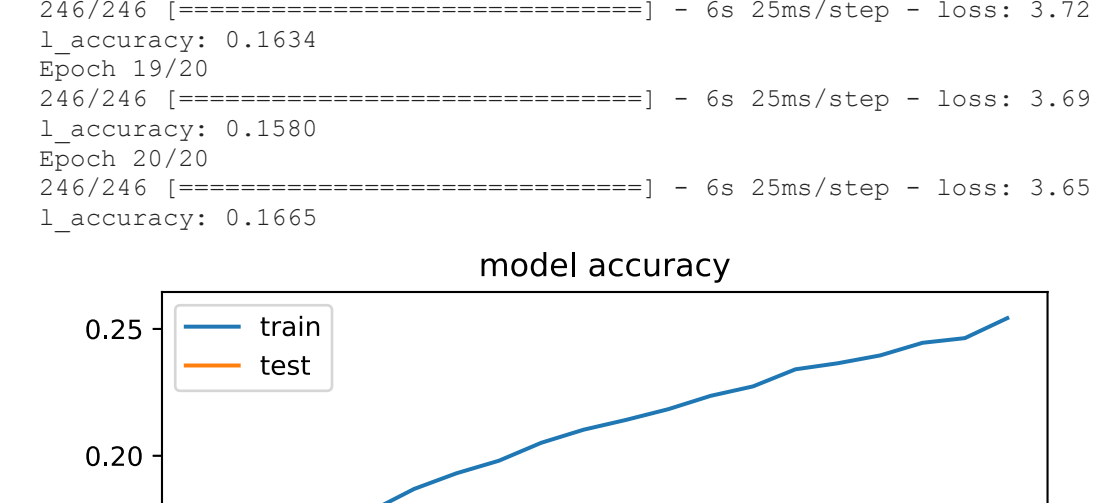
Additionally, we can see from this graph above that the time to train the model increases linearly with the number of passes completed through the neural network.

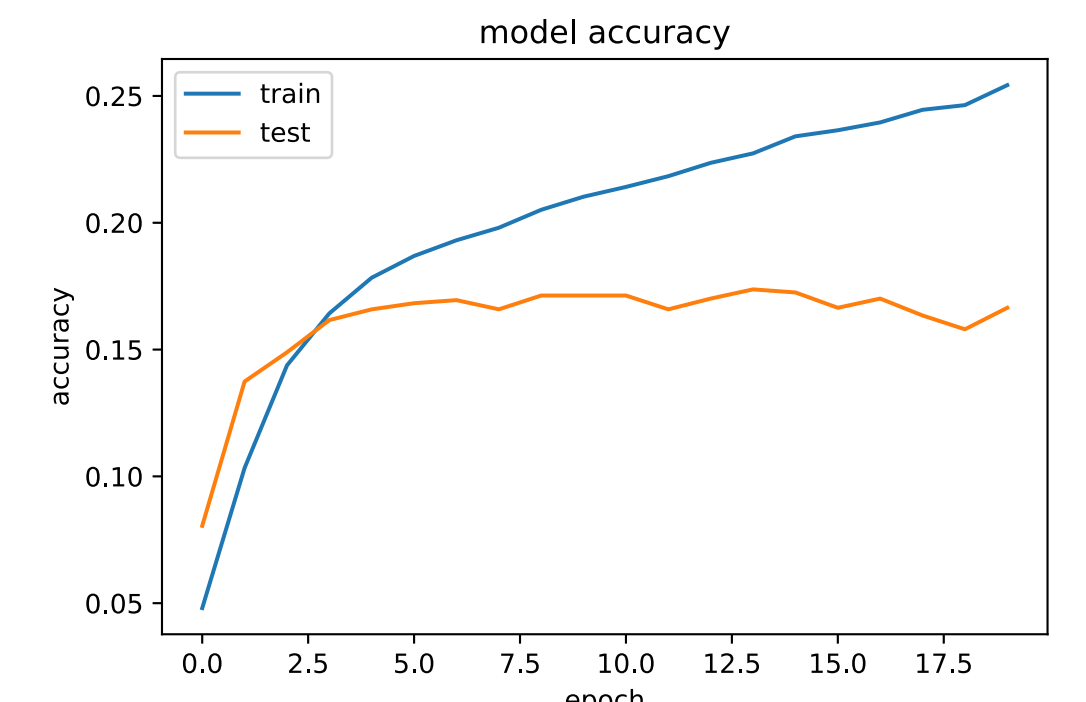
Now, we are going to use the following values for our parameters:

In [164…

```python
# The length of the song sequence that the model will look at to predict the next song.
SEQ_LENGTH = 5

# Number of nodes in the LSTM layer
LSTM_NODES = 100

# Number of epochs - how many times the layers are passed through (forward + backwards = 1 epoch)
NUM_EPOCHS = 20

# Learning Rate
LR = 0.001

create_and_evaluate_model(SEQ_LENGTH, LSTM_NODES, NUM_EPOCHS, LR)
```

```
Epoch 1/20
246/246 [==============================] - 9s 29ms/step - loss: 5.6793 - accuracy: 0.0502 - val_loss: 5.9622 - va
l_accuracy: 0.0805
Epoch 2/20
246/246 [==============================] - 6s 24ms/step - loss: 5.0932 - accuracy: 0.0865 - val_loss: 5.8209 - va
l_accuracy: 0.1374
Epoch 3/20
246/246 [==============================] - 6s 24ms/step - loss: 4.7624 - accuracy: 0.1386 - val_loss: 5.6149 - va
l_accuracy: 0.1489
Epoch 4/20
246/246 [==============================] - 6s 24ms/step - loss: 4.5648 - accuracy: 0.1598 - val_loss: 5.4905 - va
l_accuracy: 0.1616
Epoch 5/20
246/246 [==============================] - 6s 24ms/step - loss: 4.3991 - accuracy: 0.1777 - val_loss: 5.4596 - va
l_accuracy: 0.1659
Epoch 6/20
246/246 [==============================] - 6s 25ms/step - loss: 4.3095 - accuracy: 0.1835 - val_loss: 5.4483 - va
l_accuracy: 0.1683
Epoch 7/20
246/246 [==============================] - 6s 25ms/step - loss: 4.2147 - accuracy: 0.1914 - val_loss: 5.4339 - va
l_accuracy: 0.1695
Epoch 8/20
246/246 [==============================] - 6s 24ms/step - loss: 4.1575 - accuracy: 0.1963 - val_loss: 5.3296 - va
l_accuracy: 0.1659
Epoch 9/20
246/246 [==============================] - 6s 24ms/step - loss: 4.0994 - accuracy: 0.2032 - val_loss: 5.2890 - va
l_accuracy: 0.1713
Epoch 10/20
246/246 [==============================] - 6s 25ms/step - loss: 4.0377 - accuracy: 0.2118 - val_loss: 5.3262 - va
l_accuracy: 0.1713
Epoch 11/20
246/246 [==============================] - 6s 26ms/step - loss: 4.0116 - accuracy: 0.2118 - val_loss: 5.3706 - va
l_accuracy: 0.1713
Epoch 12/20
246/246 [==============================] - 6s 24ms/step - loss: 3.9403 - accuracy: 0.2242 - val_loss: 5.2758 - va
l_accuracy: 0.1659
Epoch 13/20
246/246 [==============================] - 6s 26ms/step - loss: 3.9106 - accuracy: 0.2234 - val_loss: 5.4240 - va
l_accuracy: 0.1701
Epoch 14/20
246/246 [==============================] - 6s 25ms/step - loss: 3.8898 - accuracy: 0.2271 - val_loss: 5.3879 - va
l_accuracy: 0.1737
Epoch 15/20
246/246 [==============================] - 6s 26ms/step - loss: 3.8363 - accuracy: 0.2341 - val_loss: 5.3596 - va
l_accuracy: 0.1725
Epoch 16/20
246/246 [==============================] - 6s 25ms/step - loss: 3.7861 - accuracy: 0.2424 - val_loss: 5.4190 - va
l_accuracy: 0.1665
Epoch 17/20
246/246 [==============================] - 6s 26ms/step - loss: 3.7663 - accuracy: 0.2414 - val_loss: 5.4677 - va
l_accuracy: 0.1701
Epoch 18/20
246/246 [==============================] - 6s 25ms/step - loss: 3.7258 - accuracy: 0.2465 - val_loss: 5.4435 - va
l_accuracy: 0.1634
Epoch 19/20
246/246 [==============================] - 6s 25ms/step - loss: 3.6957 - accuracy: 0.2490 - val_loss: 5.4430 - va
l_accuracy: 0.1580
Epoch 20/20
246/246 [==============================] - 6s 25ms/step - loss: 3.6553 - accuracy: 0.2594 - val_loss: 5.5645 - va
l_accuracy: 0.1665
```






```
Guessed 2856 correct out of 8217
```

When looking at the model's performance on the test set, we see that it always has a high loss and accuracy does not seem to be affected by the various parameters we modified. When runnning the model on the test set, we see that the prediction contained the actual next song 2856 out of 8217 times, for a test accuracy of ~35%.

The Setlist Generator

We will use the final model with the parameters above to create the setlist generator. We start the setlist as every set should start - by annotating the beginning of the first set. The generator uses the last LENGTH songs from the setlist (or the entire setlist if it is smaller than LENGTH), and will select songs randomly from the set of predicted songs to append to the setlist. The generator will continue to generate values until it reaches a random limit (15-25, the range of an average Phish setlist), or "END", marking the end of the show.

In [178…

```python
def generate_setlist():
    setlist = []
    rand = np.random.randint(len(songs))
    setlist.append("\"SET 1\"\n")
    idx = 1
    go = True
    limit = np.random.randint(15) + 10
    while(go and idx < limit):
        rand = np.random.randint(SEQ_LENGTH)
        if(idx < SEQ_LENGTH):
            new_songs = predict_completions(setlist[0:idx], SEQ_LENGTH)
            new_song = new_songs[rand]
        else:
            new_songs = predict_completions(setlist[idx - SEQ_LENGTH: idx + SEQ_LENGTH], SEQ_LENGTH)
            new_song = new_songs[rand]
        setlist.append(new_song)
        if(new_song == "\"END\"\n"):
            go = False
        idx += 1
    return setlist

print(generate_setlist())
```

```
['"SET 1"\n', '"Divided Sky"\n', '"Stash"\n', '"Sparkle"\n', '"I Didn\'t Know"\n', '"Split Open and Melt"\n', '"T
he Lizards"\n', '"Run Like an Antelope"\n', '"SET 2"\n', '"Wilson"\n', '"Mike\'s Song"\n', '"Simple"\n', '"Simpl
e"\n', '"Mike\'s Song"\n', '"Weekapaug Groove"\n', '"SET 2"\n', '"Mike\'s Song"\n', '"I Am Hydrogen"\n']
```

In [182…

```python
for i in range(10):
    print(generate_setlist())
```

```
['"SET 1"\n', '"Water in the Sky"\n', '"Taste"\n', '"Scent of a Mule"\n', '"Stash"\n', '"Fee"\n', '"Maze"\n', '"S
trange Design"\n', '"Run Like an Antelope"\n', '"Golgi Apparatus"\n', '"SET 3"\n', '"Also Sprach Zarathustra"\n',
'"Mike\'s Song"\n', '"Mike\'s Song"\n', '"Mike\'s Song"\n', '"Simple"\n', '"I Am Hydrogen"\n', '"Weekapaug Groov
e"\n']
['"SET 1"\n', '"Divided Sky"\n', '"Stash"\n', '"Stash"\n', '"Maze"\n', '"I Didn\'t Know"\n', '"Run Like an Antelo
pe"\n', '"SET 2"\n', '"Also Sprach Zarathustra"\n', '"Mike\'s Song"\n', '"I Am Hydrogen"\n']
['"SET 1"\n', '"Funky Bitch"\n', '"David Bowie"\n', '"Alumni Blues"\n', '"Alumni Blues"\n', '"Run Like an Antelop
e"\n', '"SET 3"\n', '"Corinna"\n', '"Alumni Blues"\n', '"The Lizards"\n', '"Mike\'s Song"\n', '"Alumni Blues"\n',
'"Letter to Jimmy Page"\n', '"END"\n']
['"SET 1"\n', '"Water in the Sky"\n', '"Scent of a Mule"\n', '"Taste"\n', '"Strange Design"\n', '"Taste"\n', '"EN
CORE"\n', '"Suzy Greenberg"\n', '"Rocky Top"\n', '"Tweezer Reprise"\n', '"ENCORE 2"\n', '"Bouncing Around the Roo
m"\n', '"Poor Heart"\n']
['"SET 1"\n', '"Divided Sky"\n', '"David Bowie"\n', '"Fee"\n', '"David Bowie"\n', '"SET 2"\n', '"AC/DC Bag"\n',
'"Divided Sky"\n', '"McGrupp and the Watchful Hosemasters"\n', '"David Bowie"\n', '"ENCORE"\n', '"Possum"\n', '"E
ND"\n']
['"SET 1"\n', '"Fee"\n', '"Fee"\n', '"I Didn\'t Know"\n', '"David Bowie"\n', '"SET 2"\n', '"AC/DC Bag"\n', '"The
Lizards"\n', '"Mike\'s Song"\n', '"Mike\'s Song"\n', '"McGrupp and the Watchful Hosemasters"\n', '"I Am Hydroge
n"\n', '"Mike\'s Song"\n']
['"SET 1"\n', '"Fee"\n', '"The Lizards"\n', '"I Didn\'t Know"\n', '"McGrupp and the Watchful Hosemasters"\n', '"D
avid Bowie"\n', '"Golgi Apparatus"\n', '"SET 3"\n', '"Wilson"\n', '"Peaches en Regalia"\n', '"You Enjoy Mysel
f"\n']
['"SET 1"\n', '"Divided Sky"\n', '"Split Open and Melt"\n', '"Maze"\n', '"Sparkle"\n', '"Stash"\n', '"SET 2"\n',
'"Maze"\n', '"Bouncing Around the Room"\n', '"Rift"\n', '"Free"\n', '"Prince Caspian"\n', '"David Bowie"\n', '"SE
T 2"\n', '"Down with Disease"\n', '"Free"\n', '"Fluffhead"\n', '"Tweezer"\n', '"The Mango Song"\n', '"Mike\'s Son
g"\n']
['"SET 1"\n', '"You Enjoy Myself"\n', '"AC/DC Bag"\n', '"Divided Sky"\n', '"You Enjoy Myself"\n', '"Fee"\n', '"Yo
u Enjoy Myself"\n', '"ENCORE"\n', '"Fire"\n', '"SET 2"\n', '"END"\n']
['"SET 1"\n', '"Water in the Sky"\n', '"Roggae"\n', '"Wolfman\'s Brother"\n', '"Limb By Limb"\n', '"Wolfman\'s Br
other"\n', '"SET 2"\n', '"Down with Disease"\n', '"Piper"\n', '"Piper"\n', '"Prince Caspian"\n', '"You Enjoy Myse
lf"\n', '"Tweezer Reprise"\n', '"END"\n']
```

Conclusion

After testing the final model against the test set, we found that the correct subsequent song was present in the list of length 5 about 35% of the time, which is significantly greater than if 5 songs were to be chosen at random from the 950 songs that are in Phish's live catalogue. However, when looking at the actual accuracy of the model on the test set, the results become less impressive, as the accuracy falls to 26%.

When looking at the setlists generated by the final function, some of them appear to be very passable as legitimate Phish setlists. However, some quirks in other entries become immediately apparent: the duplicate entries (songs are allowed to appear on a setlist two or more times (with anything greater than three being an extreme rarity), however a song will never appear twice consecutively) and the seemingly random placement of the set-indicators (causing severe inequality between sets: set 1 may contain 80% of the songs while set 2 may only have a couple of songs; not all sets made equal and sometimes one set may have significantly more songs than another set at the same show, but never to this extreme). The model's accuracy value as well as looking at the randomly generated setlists myself has led me to this conclusion: the model is competent at recognizing certain patterns in setlist and set construction (certain songs always following each other- "I am Hydrogen">"Weekapaug Groove", an encore will usually have one or two songs before show END). But, when looking at the dataset as a whole - setlist entries are too random or perhaps the dataset is not large enough to come to any reliable predictor of the bands setlists.

If I were to complete this project again, I would try to factor location and date of shows into the likelihood that a certain song will be played. As it stands now, the model is only looking for patterns in sequencing, but adding location and date data could be immensely helpful in clustering songs. Location would be useful for songs like "Fluffhead", which is performed more often at shows in New York City/State due to the lyric "Fluff Came To New York" provoking a cheer from the audience. For date, there are certain songs that were performed often in the past but infrequently now, or vice-versa (a song they may have just written, for example). These were all factors that I considered when originally brainstorming ideas for this project, however I unfortunately had to cut those aspects in order for this project to fit in a realistic time window and exist at a level of complexity that I was comfortable working on. This is a project and a topic that has inspired me, and I hope to return and improve upon it once I have more experience in the field of machine learning. Until then, only Phish can create Phish setlists.