Music Cognition

Aahana Khajanchi, Shreya Chudasama, Sreerag Mandakathil

COE, Northeastern University, Boston
khajanchi.a@jhusky.neu.edu

COE, Northeastern University, Boston
chudasama .s@jhusky.neu.edu

COE, Northeastern University, Boston
mandakathil.s@jhusky.neu.edu

Research Paper Draft

CSYE – 7245

Big-Data Systems & Intelligence Analytics

Prof. Nicholas W. Brown

COE, Northeastern University,
Boston ni.brown@northeastern.edu
April 20, 2018

ABSTRACT

The idea is to build a machine learning model that will take in a sequence of words as lyrics to a song as input and create an output which is grammatically correct and has similar writing style of a genre. For example, if we give our model a lyric of The Script song as “You can throw your hands up. You can beat the clock. You can move a mountain” we can expect lyrics for a song having lyrics like “throw your hands up and say hello” as its output. To achieve this goal, we need to build a suitable model and train that model such that it can build its own lyrics from the existing data of that band or singer. We crawled down a lyrics site (lyrics.com) and pull in the lyrics and categorized them based on artists, dominant word and sentiments. We will then scrape the data and list down the possible columns and data set. Finally, we split the content of the lyrics into chorus and verse flat files. The lyrics from the site would be in different formats but the structure would be similar so we should be able to fetch that data in csv.

1 INTRODUCTION

Can we come up with lyrics of our favorite genre? Can we train our models to write lyrics that makes sense? These questions intrigued us to create a big data machine learning algorithm to scrap lyrics from the web and to generate synthetic computer-generated lyrics for various artist.

The plan is to scrape the data from a lyrics website and serve that as input to our model, train our model to generate the lyrics similar to the lyrics fed to it. For example, if we feed our model with “I'm a 45 spinning on an old Victrola”

We expect the model to return us a song consisting of "I'm a 45 spinning on an old Victrola. I'm a two-strike swinger, I'm a Pepsi Cola"

2 METHODS-

We Create a big data machine learning algorithm to scrap lyrics from the web and to generate synthetic computer-generated lyrics for various artist, determine its billboard rating in the future, predict if the song will make on top 10 list and to display all the result in a web based application as UI.
Below is the high-level design of the process:

Fig. 1 – Process of pipeline

- Scrapping lyrics websites such as lyrics.com
  - This will be done by giving either an artist or a genre as input to a web scraper
  - Library such as Scrapy will be used to run multiple web crawlers on a single website to optimize scrapping process
- Analyzing the data and segregating the content to lyrics, artist, band, rating
  - The data will be stored in either a mongo DB or MySQL database for further processing
- To try out different machine learning models and hybrid techniques to optimize the result
- To make a web based application to interact with the system

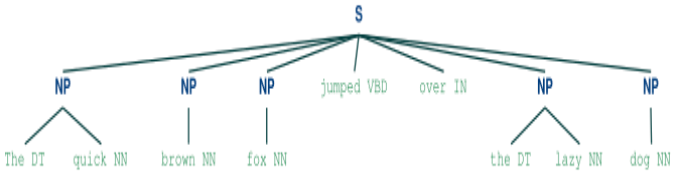
3 Model Deployment –

3.1 Pickle – It is the standard way of serializing objects in Python. We used pickle operation to serialize our machine learning algorithms and save the serialized format to a file. Later we can load this file to de-serialize our model and use it to make new predictions

4 Python Library –

4.1 NLTK

The Natural Language Toolkit, or more commonly NLTK, is a suite of libraries and programs for symbolic and statistical natural language processing (NLP) in the Python programming language. NLTK includes graphical demonstrations and sample data. NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum.



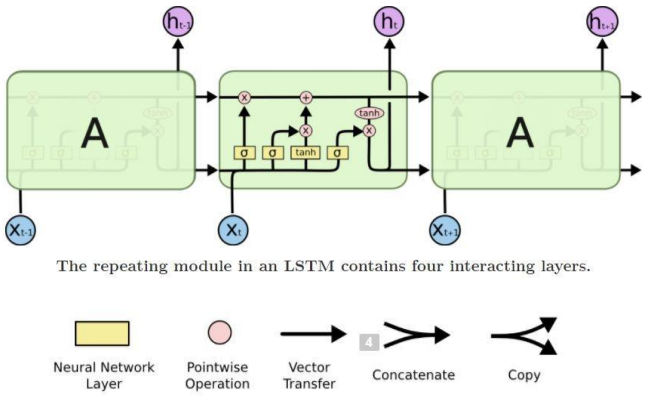
5.3 Django

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

5.4 LSTM

Long Short-Term Memory networks – usually just called "LSTMs" – are a special kind of RNN, capable

of learning long-term dependencies. LSTMs also have this chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way.



The repeating module in an LSTM contains four interacting layers.

The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates.Gates are a way to optionally let information through. They are composed out of a sigmoid neural net layer and a pointwise multiplication operation.

The sigmoid layer outputs numbers between zero and one, describing how much of each component should be let through. A value of zero means “let nothing through,” while a value of one means “let everything through!”. LSTM follows a step by step procedure so remembering information for long periods of time is practically their default behavior, not something they struggle to learn.

5 Code with Documentation -

5.1 Web Scraping – To scrap the links from all website.

```
album_link

[('Rock',
  'Little Anthony & The Imperials',
  'I Love Music - Only Original Recondings Album',
  'http://www.songlyrics.com/little-anthony-the-imperials/i-love-music-only-original-recondings/'),
 ('Rock',
  'Twenty One Pilots',
  'Blurryface Album',
  'http://www.songlyrics.com/twenty-one-pilots/blurryface/'),
 ('Rock',
  'Panic! At the Disco',
  'Death of a Bachelor Album',
  'http://www.songlyrics.com/panic!-at-the-disco/death-of-a-bachelor/'),
 ('Rock',
  'Metallica',
  'Hardwired...to Self-Destruct Album',
  'http://www.songlyrics.com/metallica/hardwired-to-self-destruct/'),
 ('Rock',
  'Soundtrack',
  'Suicide Squad: The Album OST Album',
  'http://www.songlyrics.com/soundtrack/suicide-squad-the-album-ost/'),
 ('Rock',
  'Twenty One Pilots',
  'Vessel Album',
  'http://www.songlyrics.com/twenty-one-pilots/vessel/'),
```

Converting it into a dataframe and scraping lyrics out of each link.

```
lyric = lyrics.find(id='songlyricsDiv').get_text()

lyric

"Well, I'm alright, alright\n\t'm alright, yes, I'm alright\nsince my baby told me she love me\n\t'm alright, I'm alright\nsince she told me, she put no one above me\n\t'
```

Cleaning the data frame and removing duplicate data

```
pop_d.drop_duplicates('Song')
```

6	Rock	Little Anthony & The Imperials	I Love Music - Only Original Recondings	Theme from	Sorry, we have no Little Anthony & The Imperia...
7	Rock	Little Anthony & The Imperials	I Love Music - Only Original Recondings	You're Gonna Live Forever in Me	Sorry, we have no Little Anthony & The Imperia...
8	Rock	Twenty One Pilots	Blurryface	Fairly Local	I'm fairly local, I've been around\n\n\t've seen ...
9	Rock	Twenty One Pilots	Blurryface	Tear In My Heart	Sometimes you've got to bleed to know,\n\nThat y...
10	Rock	Twenty One Pilots	Blurryface	Lane Boy	They say "stay in your lane boy, lane boy"\nBu...
11	Rock	Twenty One Pilots	Blurryface	Ride	I just wanna stay in the sun where I find\n\t k...
12	Rock	Twenty One Pilots	Blurryface	Stressed Out	I wish I found some better sounds no one's eve...
13	Rock	Twenty One Pilots	Blurryface	We Don't Believe What's on TV	Yeah, yeah, yeah\n\nWe don't believe what's on T...
14	Rock	Twenty One Pilots	Blurryface	Heavydirtysoul	There's an infestation in my mind's imaginatio...
15	Rock	Twenty One Pilots	Blurryface	The Judge	Na Na Na Na Oh Oh\r\nNa Na Na Na Oh Oh\r\nNa N...

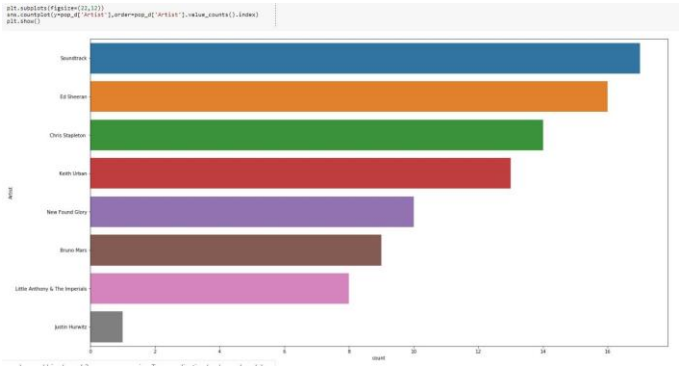
	Genre	Artist	Album	Song	Lyrics
0	pop	Ed Sheeran	÷ (Divide)	Eraser	I was born inside a small town, I lost that st...
1	pop	Ed Sheeran	÷ (Divide)	Castle on the Hill	When I was six years old I broke my leg\r\n\t w...
2	pop	Ed Sheeran	÷ (Divide)	Dive	Maybe I came on too strong\r\n\nMaybe I waited t...
3	pop	Ed Sheeran	÷ (Divide)	Shape of You	The club isn't the best place to find a lover\...
4	pop	Ed Sheeran	÷ (Divide)	Perfect	I found a love for me\r\n\nDarling, just dive ri...

Storing the data in a Json file.

```
[ ]  pop_d.to_json('songData.json',orient='records')
```

```
for k in pop_d[pop_d['Genre'] == "Pop"]["Lyrics"]:
    file = open('Pop.txt', 'a')
    file.write(k)
    file.close()
```

Plotting the graph to know number of songs of few artist.



```
#create sequences
sequences = []
next_words = []
for i in range(0, len(x_text) - seq_length, sequences_step):
    sequences.append(x_text[i: i + seq_length])
    next_words.append(x_text[i + seq_length])

print('nb sequences:', len(sequences))


print('Vectorization.')
X = np.zeros((len(sequences), seq_length, vocab_size), dtype=np.bool)
y = np.zeros((len(sequences), vocab_size), dtype=np.bool)
for i, sentence in enumerate(sequences):
    for t, word in enumerate(sentence):
        X[i, t, vocab[word]] = 1
    y[i, vocab[next_words[i]]] = 1
```

```
nb sequences: 10105
Vectorization.
```

Training the Model

Creating directories and assigning space and building a vocabulary for the model

```
# data_dir = 'data/text.txt'# data directory containing input.txt
save_dir = 'save' # directory to store models
rnn_size = 128 # size of RNN
batch_size = 30 # minibatch size
seq_length = 15 # sequence length
num_epochs = 30 # number of epochs
learning_rate = 0.001 #learning rate
sequences_step = 3 #step to create sequences
```

```
vocab_file = save_dir+ "/words_vocab.pkl"
vocab_file
```

```
'save/words_vocab.pkl'
```

```
# lyrics = pop_d['Lyrics']
lyrics = pop_d[pop_d['Genre'] == "Pop"]["Lyrics"]
test2 = lyrics.str.cat(sep='\n')
test = lyrics.str.cat(sep='\n')
test = test.replace('\n', ' . ')
print(test)
```

```
. Habit of you . I'm a 45 spinning on an old Victrola . I'm a two strike swinger, I'm a Pepsi Cola .
```

Counting the number of times, the word has been used

```
word_counts = collections.Counter(x_text)
word_counts
```

```
Counter({'I': 1033,
         'was': 89,
         'born': 3,
         'inside': 13,
         'a': 396,
         'small': 3,
         'town,': 2,
         'lost': 12,
         'that': 242,
         'state': 5,
         'of': 227,
         'mind': 28,
         '.': 4502,
         'Learned': 1,
         'to': 454,
         'sing': 14,
         'the': 838,
         "lord's": 1,
         'house': 9,
         'But': 156,
         'stopped': 2,
         'at': 54,
         'age': 2,
         'nine': 2,
         'forget': 4,
```

Creating Sequences

Building LSTM model and calculating Epoch and loss on our training data

Epoch: A full pass over all of your training data

Loss: A scalar value that we attempt to minimize during our training of the model. The lower the loss, the closer our predictions are to the true labels.

Adam is a combination of the advantages of two other extensions of stochastic gradient descent. Specifically:

Adaptive Gradient Algorithm (AdaGrad) that maintains a per-parameter learning rate that improves performance on problems with sparse gradients (e.g. natural language and computer vision problems).

Root Mean Square Propagation (RMSProp) that also maintains per-parameter learning rates that are adapted based on the average of recent magnitudes of the gradients for the weight (e.g. how quickly it is changing). This means the algorithm does well on online and non-stationary problems (e.g. noisy).

Adam realizes the benefits of both AdaGrad and RMSProp.

```
# build the model: a single LSTM
print('Build LSTM model.')
model = Sequential()
model.add(LSTM(rnn_size, input_shape=(seq_length, vocab_size)))
model.add(Dense(vocab_size))
model.add(Activation('softmax'))
```

```
Build LSTM model.
```

```
#adam optimizer
optimizer = Adam(lr=learning_rate)
model.compile(loss='categorical_crossentropy', optimizer=optimizer)
```

```
#fit the model
model.fit(X, y,batch_size=batch_size,epochs=num_epochs)

#save the model
model.save(save_dir + "/" + 'my_model.h5')
```

```
Epoch 1/30
10105/10105 [==============================] - 18s 2ms/step - loss: 6.3398
Epoch 2/30
10105/10105 [==============================] - 18s 2ms/step - loss: 5.8088
Epoch 3/30
10105/10105 [==============================] - 23s 2ms/step - loss: 5.7108
Epoch 4/30
10105/10105 [==============================] - 21s 2ms/step - loss: 5.6133
```

```
Epoch 26/30
10105/10105 [==============================] - 19s 2ms/step - loss: 1.0492:
Epoch 27/30
10105/10105 [==============================] - 18s 2ms/step - loss: 0.9264
Epoch 28/30
10105/10105 [==============================] - 18s 2ms/step - loss: 0.8062
Epoch 29/30
10105/10105 [==============================] - 18s 2ms/step - loss: 0.6978
Epoch 30/30
10105/10105 [==============================] - 18s 2ms/step - loss: 0.5971
```

So an epoch concludes when it has finished a training pass over all 10105 of the observations.

As we can see in the above observation, there is a decrease in loss over 30 number of epochs. It has reduced to 0.59 from 6.33.

Building the model using LSTM Keras

```
# load the model
print("loading model...")
model = load_model('my_model.h5')
```

```
loading model...
```

```
def sample(preds, temperature=1.0):
    # helper function to sample an index from a probability array
    preds = np.asarray(preds).astype('float64')
    preds = np.log(preds) / temperature
    exp_preds = np.exp(preds)
    preds = exp_preds / np.sum(exp_preds)
    probas = np.random.multinomial(1, preds, 1)
    return np.argmax(probas)
```

Methods for Python Text to Speech-

GetProperty(name : string) – gets the cu rrent value of an engine property.

SetProperty - Queues a command to set an engine property.

The new property value affects all utterances queued after this command.

Say() - Queues a command to speak an utterance. The speech is output according to the properties set before this command in the queue.

engine.runAndWait()- Blocks while processing all currently queued commands. Invokes callbacks for engine notifications appropriately. Returns when all commands queued before this call are emptied from the queue.

```
# Using Python Text To Speech
import pyttsx3
engine = pyttsx3.init()
voices = engine.getProperty('voices')  # Gets the current value of an engine property.

rate = engine.getProperty('rate')
engine.setProperty('voice', voices[0].id)  # Queues a command to set an engine property.
engine.say(test3)  # Queues a command to speak an utterance. The speech is output accordi

engine.runAndWait()  # Blocks while processing all currently queued commands. Invokes cal
```

Generating Lyrics

Below is the code that generates lyrics by calculating and predicting the next word

```
[ ]  generated = ''
     sentence = []
     for i in range (seq_length):
         sentence.append("a")

     seed = seed_sentences.split()

     for i in range(len(seed)):
         sentence[seq_length-i-1]=seed[len(seed)-i-1]

     generated += ' '.join(sentence)
     print('Generating text with the following seed: "' + ' '.join(sentence) + '"')



     #generate the text
     for i in range(words_number):
         #create the vector
         x = np.zeros((1, batch_size, vocab_size))
     #     print(x.shape)

         for t, word in enumerate(sentence):
     #         print(x[0, t, vocab[word]])
             x[0, t, vocab[word]] = 1.
     #     print(x.size)
         #calculate next word
         preds = model.predict(x, verbose=0)[0]
         next_index = sample(preds, 1)
         next_word = vocabulary_inv[next_index]

         #add the next word to the text
         generated += " " + next_word
         # shift the sentence by one, and and the next word at its end
         sentence = sentence[1:] + [next_word]

     # print()
     # print("Generated Lyrics:")
     # print(generated)

     # print()
```

```
Generating text with the following seed: "a a a a a a a a a a a a Love me like"
```

Printing out the lyrics generated using the model

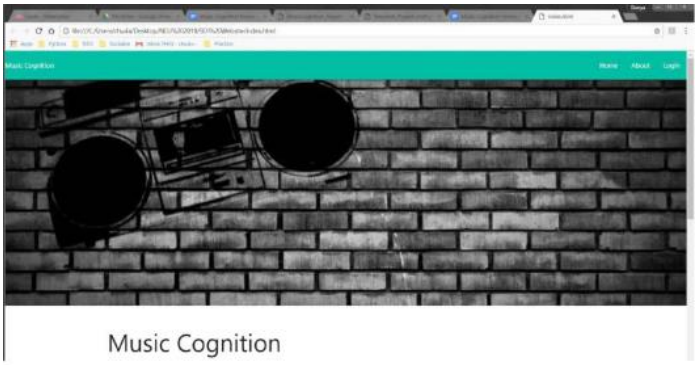
```
print("Generated Lyrics:")
test3 = generated.replace('. .','\n\n')
test3 = test3.replace(' . ','\n')
print(test3)
```

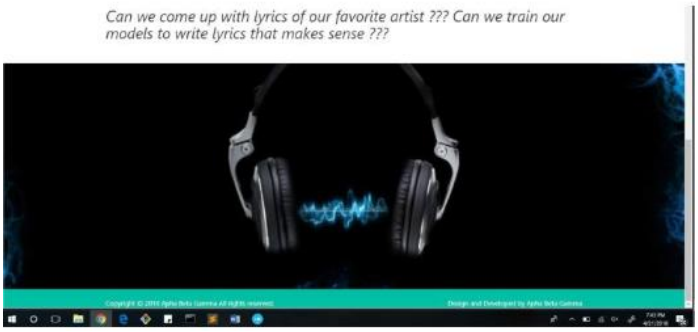
```
Generated Lyrics:
a a a a a a a a a a a a a a a start world, edges at
So need gotta can for me

 I'd have to the crashed up the mind
Your the sitting time with thought this behind
My get are own pain 3: open
She kiss the castle but the night
Cause I don’t everything I blame Alicia

 buy the truck, karat) cheapest lost
For are out
Wo excited Ryan pray
This someone our on about else
While one doesn’t tryna fall
But I was broken wide of my is inside I'm used
But your up is a anymore right Cougar,
While bet a habit oh, says, I'm still looking it
And every start to find
And 1] I Pop it
```

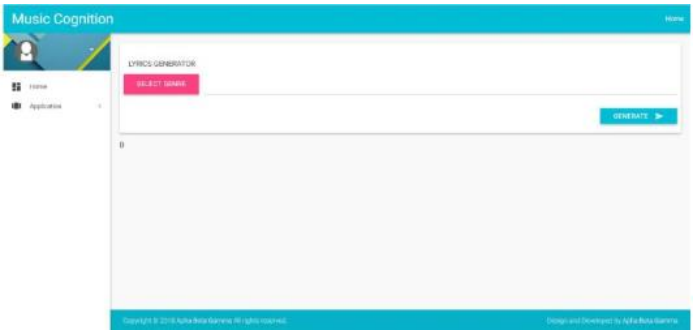
As you can see, the above sentences does not make much sense. We have used **markovify** to generate sentences that makes sense and are grammar correct.

```
import markovify
from gtts import gTTS
text_model = markovify.NewlineText(str(test2))
s = ""
# Print five randomly-generated sentences
for i in range(5):
    o = text_model.make_sentence()
    if(o!=None):
        print(o)
        s += ", "+o
print(s)
```

```
While everyone is living it up and down
I'm looking for a fresh start?
Are you living or just throw stones in?
We push and pull like a wild man outta the city
For a good time tonight, girl
, While everyone is living it up and down, I'm looking for a fresh start?,
```







6 Results – After implementing and testing the above models to generate lyrics below is the generated text.

```
Self-inflicted, but we had it made
Would you pray for peace
Well I found a woman, stronger than anyone I know without it's no fun
Well here I am found
But I guess you look perfect tonight
```

Our best model is made using LSTM Neural Network and Markov chain rule.

ACKNOWLEDGMENTS

Authors would like to thank our professor, teaching assistants, family and friends who supported in the completion of this research project. Appreciating everyone who helped us knowingly or unknowingly for this project.

Reference

1. http://colah.github.io/posts/2015-08-Understanding-LSTMs/
2. https://towardsdatascience.com/understanding-lstm-and-its-quick-implementation-in-keras-for-sentim ent-analysis-af410fd85b47
3. https://medium.com/@ivanliljeqvist/using-ai-to-generate-lyrics-5aba7950903
4. http://www.nltk.org/book/ch01.html
5. https://nlp.stanford.edu/courses/cs224n/2009/fp/5.pdf