8A. Song Lyrics Generator

March 11, 2019

1 8A. Song Lyrics Generator

In this lab, you will scrape a website to get lyrics of songs by your favorite artist. Then, you will train a model called a Markov chain on these lyrics so that you can generate a song in the style of your favorite artist.

2 Question 1. Scraping Song Lyrics

Find a web site that has lyrics for several songs by your favorite artist. Scrape the lyrics into a Python list called `lyrics`, where each element of the list represents the lyrics of one song.

Tips: - Find a web page that has links to all of the songs, like this one. [*Note:* It appears that `azlyrics.com` blocks web scraping, so you'll have to find a different lyrics web site.] Then, you can scrape this page, extract the hyperlinks, and issue new HTTP requests to each hyperlink to get each song. - Use `time.sleep()` to stagger your HTTP requests so that you do not get banned by the website for making too many requests.

```
In [1]: import requests
        import time

        from bs4 import BeautifulSoup

        names = ["gods-plan", "hotline-bling", "hold-on-were-going-home", "one-dance",
                 "in-my-feelings", "nice-for-what", "find-your-love",
                 "too-good", "best-i-ever-had", "passionfruit"]

        lyrics = []

        for i in names:
            resp = requests.get("http://www.metrolyrics.com/{0}-lyrics-drake.html".format(i))
            time.sleep(.5)
            soup = BeautifulSoup(resp.content)

            song = soup.find_all("p", attrs={"class":"verse"})

            final_lyrics = ""
```

```
            time.sleep(.25)
            for i in range(len(song)):
                verse = song[i].text.strip()
                #print(verse)
                #final_lyrics = " <N> " + verse
                final_lyrics += verse

            lyrics.append(final_lyrics)
```

```
In [2]: lyrics[0] = lyrics[0].replace('[Intro]', "")
        lyrics[0] = lyrics[0].replace('[Chorus 1]', "")
        lyrics[0] = lyrics[0].replace('[Chorus 2]', "")
        lyrics[0] = lyrics[0].replace('[Verse 1]', "")
        lyrics[0] = lyrics[0].replace('[Verse 2]', "")
        lyrics[0] = lyrics[0].replace('[Bridge]', "")
        for i in range(len(lyrics)):
            lyrics[i] = lyrics[i].replace('\n', " <N> ")
```

`pickle` is a Python library that serializes Python objects to disk so that you can load them in later.

```
In [3]: import pickle
        pickle.dump(lyrics, open("lyrics.pkl", "wb"))
```

3 Question 2. Unigram Markov Chain Model

You will build a Markov chain for the artist whose lyrics you scraped in Lab A. Your model will process the lyrics and store the word transitions for that artist. The transitions will be stored in a dict called `chain`, which maps each word to a list of "next" words.

For example, if your song was "The Joker" by the Steve Miller Band, `chain` might look as follows:

```
chain = {
    "some": ["people", "call", "people"],
    "call": ["me", "me", "me"],
    "the": ["space", "gangster", "pompitous", ...],
    "me": ["the", "the", "Maurice"],
    ...
}
```

Besides words, you should include a few additional states in your Markov chain. You should have `"<START>"` and `"<END>"` states so that we can keep track of how songs are likely to begin and end. You should also include a state called `"<N>"` to denote line breaks so that you can keep track of where lines begin and end. It is up to you whether you want to include normalize case and strip punctuation.

So for example, for "The Joker", you would add the following to your chain:

```
chain = {
```

```
    "<START>": ["Some", ...],
    "Some": ["people", ...],
    "people": ["call", ...],
    "call": ["me", ...],
    "me": ["the", ...],
    "the": ["space", ...],
    "space": ["cowboy,", ...],
    "cowboy,": ["yeah", ...],
    "yeah": ["<N>", ...],
    "<N>": ["Some", ..., "Come"],
    ...,
    "Come": ["on", ...],
    "on": ["baby", ...],
    "baby": ["and", ...],
    "and": ["I'll", ...],
    "I'll": ["show", ...],
    "show": ["you", ...],
    "you": ["a", ...],
    "a": ["good", ...],
    "good": ["time", ...],
    "time": ["<END>", ...],
}
```

Your chain will be trained on not just one song, but by all songs by your artist.

```
In [4]: def train_markov_chain(lyrics):
            """
            Args:
              - lyrics: a list of strings, where each string represents
                        the lyrics of one song by an artist.

            Returns:
              A dict that maps a single word ("unigram") to a list of
              words that follow that word, representing the Markov
              chain trained on the lyrics.
            """
            chain = {"<START>" : []}

            for lyric in lyrics:
                count = 0
                #print(lyric)
                words = lyric.split()
                #print(words)
                chain["<START>"].append(words[0])
                for i in range(len(words)):
                    if i < len(words) - 1:
                        if words[i] in chain:
                            chain[words[i]].append(words[i+1])
```

```
                    else:
                        chain[words[i]] = [words[i+1]]
                else:
                    if words[i] in chain:
                        chain[words[i]].append("<END>")
                        #chain[words[i]].append("END")
                    else:
                        chain[words[i]] = ["<END>"]
                        #chain[words[i]] = "END"

        return chain
```

In [5]:
```
# Load the pickled lyrics object that you created in Lab A.
import pickle
lyrics = pickle.load(open("lyrics.pkl", "rb"))

# Call the function you wrote above.
chain = train_markov_chain(lyrics)

# What words tend to start a song (i.e., what words follow the <START> tag?)
print(chain["<START>"])

# What words tend to begin a line (i.e., what words follow the line break tag?)
print(chain["<N>"][:20])
```

```
['<N>', 'You', 'I', 'Baby,', 'Trap,', 'I', "I'm", 'Oh', 'You', 'Hold']
['Yeah', 'They', 'I', 'Tryna', "Don't", 'You', 'I', 'Guess', 'Hope', 'They', "God's", 'I', 'I'
```

Now, let's generate new lyrics using the Markov chain you constructed above. To do this, we'll begin at the `"<START>"` state and randomly sample a word from the list of words that follow `"<START>"`. Then, at each step, we'll randomly sample the next word from the list of words that followed each current word. We will continue this process until we sample the `"<END>"` state. This will give us the complete lyrics of a randomly generated song!

You may find the `random.choice()` function helpful for this question.

In [6]:
```
import random

def generate_new_lyrics(chain):
    """
    Args:
      - chain: a dict representing the Markov chain,
               such as one generated by generate_new_lyrics()

    Returns:
      A string representing the randomly generated song.
    """

    # a list for storing the generated words
```

```
    words = []
    # generate the first word
    words.append(random.choice(chain["<START>"]))

    # YOUR CODE HERE
    done = False
    while not done:
        words.append(random.choice(chain[words[len(words) - 1]]))
        if words[len(words) - 1] == "<END>":
            done = True


    # join the words together into a string with line breaks
    lyrics = " ".join(words[:-1])
    return "\n".join(lyrics.split("<N>"))
```

```
In [7]: print(generate_new_lyrics(chain))
```

```
I just being right
 It feels like you always stay at me
 Give to ask you my hands around me
 Just hold onI keep letting you need a patient in your lovin'
 Wonder if you're okay
 Last night I need that hotline bling
 Why won't mention
 You're everything
 I hope that you the fuckin' best
 Got a realization
 Said you'd die for granted
 That they wishin'
 And when that ass back up
 Just hold on Southside G, yu
 That can I can't get the rumors and they wishin'
 She say, "Do you the friendship
 Are you see if I ever leave
 First, last slice
 I bet if this forever
 I'll be real one, in your heart
 KB, do you know you been peepin' what they wishin'
 Hangin' with you
 First, last slice
 Just make it peaceful is a one
 I get rightI promise that ass
 You take itI'm too good to call me
 'Cause if you're a nigga hit it, yupKiki, do it
 You ain't even know it taking a G.O.D., yeah, wait a real big
 All up and I promise that I need my love my bed and me, care for me!
 And I'm more time I can only mean one thing every single time I left your lovin'
```

You said you'd die for granted
Watch the fact that...I'm too good girl and I won't mention
Give to get a hold on my eyes on me to me!
There for me!
Wonder if I better find your expectations
No, I'm away
You the new me down pon it from the only mean one that shit
You said you'd die for me!
Hey, hey, hey
Cock up on my heart
And when friends always touching road
Then nothing's gonna tear us apart
Going places where do you
Every time 'fore I need ya
I just leave
You hear but you everything
I get some Henny
You act so long
Doing overtime for me! (A song
I want ya, and wishin' and they running out commitment for me!
I know you
Back up, back up on we're going home
Hey, hey, hey
Said you'd die for you used to these niggas
I can't end like I ever hadBest I think there's never alone (things alone)
That can spend whatever on me
I can't miss
I'll be forgotten
They wishin'
I give to call me is costing me
I need ya
I hope you know shorty and wishin' and I ever had
Are you riding say you'll never run away
You don't get em gassed up
Have a hold on we're going home (home)I got some brothers that hotline bling
I do you my wallet
Last night, I don't understand itYears go
Higher powers taking a one dance
You the safe
Gotta make your mothafuckin' roll on my hand
Back up, back up the same thing
Young money ain't ever had
It's a Hennessy in
Feelin' for me, 'cause I get over backwards for it
Passive with me feel like you
It's hard at buildin' trust from miles away
You take my momma, I'm way too good to you are...
I think I just gotta be there for a hold on Southside G, ay, road, ay

```
High school pics, you don't understand it like the fuckin' best
'Cause we're going home (home)I got to you andI need my love for me, care for me, cry for me?
Why won't you ain't ever had
Passive with me when its stopping
And when that ass back in so accustomed to the key under the fuckin' bra strap pop
There for my love and wishin' and me, we gon' live for now
Got a distance
They wishin' and wishin' and I just be over you the same thing every single show she out regul
I ever wanted
You said you'd be over backwards for me?!Gotta hit her up on me
I'm down
You act so official
You used to somebody
I thought you know alot of pages in my patience
I can't blame you, no, no me things
Oti, oti, there's nothing left to you the city, you know it for me!
They gon' live it tatted on Southside G, ay, ay
Then nothing's gonna tear us apartToo many times
You know it up a mission
Mi wi give to my love me
I don't see the one thing to you to me!
There for it up on
Wonder if you're down pon it tatted on it like it
B-bring that hotline bling
I had
I better find your slot 'til a mission
Then nothing's gonna tear us apartToo many times I've been inside, know too and I guess being
Thinkin' my love
I swear I give to the fuckin' best
I give it real with me
Said you'd be there as you need a blank disc
I'll be no lover in here tonight
And I'm down
Why you
```

4 Question 3. Bigram Markov Chain Model

Now you'll build a more complex Markov chain that uses the last *two* words (or bigram) to predict the next word. Now your dict `chain` should map a *tuple* of words to a list of words that appear after it.

As before, you should also include tags that indicate the beginning and end of a song, as well as line breaks. That is, a tuple might contain tags like `"<START>"`, `"<END>"`, and `"<N>"`, in addition to regular words. So for example, for "The Joker", you would add the following to your chain:

```
chain = {
    (None, "<START>"): ["Some", ...],
```

```
    ("<START>", "Some"): ["people", ...],
    ("Some", "people"): ["call", ...],
    ("people", "call"): ["me", ...],
    ("call", "me"): ["the", ...],
    ("me", "the"): ["space", ...],
    ("the", "space"): ["cowboy,", ...],
    ("space", "cowboy,"): ["yeah", ...],
    ("cowboy,", "yeah"): ["<N>", ...],
    ("yeah", "<N>"): ["Some", ...],
    ("time", "<N>"): ["Come"],
    ...,
    ("<N>", "Come"): ["on", ...],
    ("Come", "on"): ["baby", ...],
    ("on", "baby"): ["and", ...],
    ("baby", "and"): ["I'll", ...],
    ("and", "I'll"): ["show", ...],
    ("I'll", "show"): ["you", ...],
    ("show", "you"): ["a", ...],
    ("you", "a"): ["good", ...],
    ("a", "good"): ["time", ...],
    ("good", "time"): ["<END>", ...],
}
```

```
In [8]: def train_markov_chain(lyrics):
            """
            Args:
              - lyrics: a list of strings, where each string represents
                        the lyrics of one song by an artist.

            Returns:
              A dict that maps a tuple of 2 words ("bigram") to a list of
              words that follow that bigram, representing the Markov
              chain trained on the lyrics.
            """
            chain = {(None, "<START>"): []}
            for lyric in lyrics:
                # YOUR CODE HERE
                count = 0
                #print(lyric)
                words = lyric.split()
                #print(words)
                chain[(None, "<START>")].append(words[0])
                chain[("<START>", words[0])] = [(words[1])]
                for i in range(len(words)-1):
                    if i < len(words) - 2:
                        if words[i] in chain and words[i+1] in chain:
                            chain[(words[i], words[i+1])].append(words[i+2])
                        else:
```

```
                chain[(words[i], words[i+1])] = [words[i+2]]
            else:
                if words[i] in chain and words[i+1] in chain:
                    chain[(words[i], words[i+1])].append("<END>")
                    #chain[words[i]].append("END")
                else:
                    chain[(words[i], words[i+1])] = ["<END>"]
                    #chain[words[i]] = "END"

    return chain
```

In [9]:
```
# Load the pickled lyrics object that you created in Lab A.
import pickle
lyrics = pickle.load(open("lyrics.pkl", "rb"))

# Call the function you wrote above.
chain = train_markov_chain(lyrics)

# What words tend to start a song (i.e., what words follow the <START> tag?)
print(chain[(None, "<START>")])
```

```
['<N>', 'You', 'I', 'Baby,', 'Trap,', 'I', "I'm", 'Oh', 'You', 'Hold']
```

Now, let's generate new lyrics using the Markov chain you constructed above. To do this, we'll begin at the `(None, "<START>")` state and randomly sample a word from the list of words that follow this bigram. Then, at each step, we'll randomly sample the next word from the list of words that followed the current bigram (i.e., the last two words). We will continue this process until we sample the `"<END>"` state. This will give us the complete lyrics of a randomly generated song!

In [10]:
```
import random

def generate_new_lyrics(chain):
    """
    Args:
      - chain: a dict representing the Markov chain,
               such as one generated by generate_new_lyrics()

    Returns:
      A string representing the randomly generated song.
    """

    # a list for storing the generated words
    words = []
    # generate the first word
    words.append(random.choice(chain[(None, "<START>")]))
    words.append(random.choice(chain[("<START>", words[0])]))
```

```
            # YOUR CODE HERE
            done = False
            while not done:
                words.append(random.choice(chain[(words[len(words)-2], words[len(words)-1])]))

                #if words[len(words) - 1] == "the":
                if words[len(words) - 1] == "<END>":
                    done = True


            # join the words together into a string with line breaks
            lyrics = " ".join(words[:-1])
            return "\n".join(lyrics.split("<N>"))
```

```
In [11]: print(generate_new_lyrics(chain))
```

```
I'm more than it seems
 Sacrificing things
 And I say you the f...
 Uh
 Oh yeah
 Tension between us just like picket fences
 You got issues that I won't mention for now
 They keep fallin' apartPassionate from miles away
 Passive with the things you say
 Passin' up on my old ways
 I can't blame you, no, noTrying to think of the right thing to say
```

5 Analysis

Compare the quality of the lyrics generated by the unigram model (in Lab B) and the bigram model (in Lab C). Which model seems to generate more reasonable lyrics? Can you explain why? What do you see as the advantages and disadvantages of each model?

The lyrics generated form the unigram model are not as fluid as the bigram model. The lyrics make more sense in the bigram model because it is using pairs of words so then there will be more unique keys. Then for each key, the amount of values to sample from will be smaller. Therefore, the bigram will produce lyrics that read easier. However, one disadvantage to this is that we lose the "random" effect, and predicting the lyrics might be easier if the chain starts off with unique tuples of keys. This in turn leads to actual repeated lyrics from the artist's songs. For example, in the bigram above, lines 8 and 9 are word for word with Drake's song "Passionfruit".

6 Submission Instructions

Once you are finished, follow these steps:

1. Restart the kernel and re-run this notebook from beginning to end by going to `Kernel > Restart Kernel and Run All Cells`.

2. If this process stops halfway through, that means there was an error. Correct the error and repeat Step 1 until the notebook runs from beginning to end.
3. Double check that there is a number next to each code cell and that these numbers are in order.

Then, submit your lab as follows:

1. Go to `File > Export Notebook As > PDF`.
2. Double check that the entire notebook, from beginning to end, is in this PDF file. (If the notebook is cut off, try first exporting the notebook to HTML and printing to PDF.)
3. Upload the PDF to PolyLearn.