Text Generation

Introduction

Markov chains can be used for very basic text generation. Think about every word in a corpus as a state. We can make a simple assumption that the next word is only dependent on the previous word - which is the basic assumption of a Markov chain.

Markov chains don't generate text as well as deep learning, but it's a good (and fun!) start.

Select Text to Imitate

In this notebook, we're specifically going to generate text in the style of Ali Wong, so as a first step, let's extract the text from her comedy routine.

In [1]:
```
# Read in the corpus, including punctuation!
import pandas as pd

data = pd.read_pickle('corpus.pkl')
data
```

Out[1]:

	transcript	full_name
ali	Ladies and gentlemen, please welcome to the st...	Ali Wong
anthony	Thank you. Thank you. Thank you, San Francisco...	Anthony Jeselnik
bill	[cheers and applause] All right, thank you! Th...	Bill Burr
bo	Bo What? Old MacDonald had a farm E I E I O An...	Bo Burnham
dave	This is Dave. He tells dirty jokes for a livin...	Dave Chappelle
hasan	[theme music: orchestral hip-hop] [crowd roars...	Hasan Minhaj
jim	[Car horn honks] [Audience cheering] [Announce...	Jim Jefferies
joe	[rock music playing] [audience cheering] [anno...	Joe Rogan
john	All right, Petunia. Wish me luck out there. Yo...	John Mulaney
louis	Intro\nFade the music out. Let's roll. Hold th...	Louis C.K.
mike	Wow. Hey, thank you. Thanks. Thank you, guys. ...	Mike Birbiglia
ricky	Hello. Hello! How you doing? Great. Thank you....	Ricky Gervais

In [2]:
```
# Extract only Ali Wong's text
ali_text = data.transcript.loc['ali']
ali_text[:200]
```

Out[2]:
```
'Ladies and gentlemen, please welcome to the stage: Ali Wong! Hi. Hello! Welcome! Thank you! Thank you for coming. Hello! Hello. We are gonna have to
get this shit over with, ’cause I have to pee in, l'
```

Build a Markov Chain Function

We are going to build a simple Markov chain function that creates a dictionary:

- The keys should be all of the words in the corpus
- The values should be a list of the words that follow the keys

In [3]:

```python
from collections import defaultdict

def markov_chain(text):
    '''The input is a string of text and the output will be a dictionary with each word as
       a key and each value as the list of words that come after the key in the text.'''

    # Tokenize the text by word, though including punctuation
    words = text.split(' ')

    # Initialize a default dictionary to hold all of the words and next words
    m_dict = defaultdict(list)

    # Create a zipped list of all of the word pairs and put them in word: list of next words format
    for current_word, next_word in zip(words[0:-1], words[1:]):
        m_dict[current_word].append(next_word)

    # Convert the default dict back into a dictionary
    m_dict = dict(m_dict)
    return m_dict
```

In [4]:

```python
# Create the dictionary for Ali's routine, take a look at it
ali_dict = markov_chain(ali_text)
ali_dict
```

Out[4]:

```
{'Ladies': ['and'],
 'and': ['gentlemen,',
  'foremost,',
  'then',
  'have',
  'there’s',
  'resentment',
  'get',
  'get',
  'says,',
  'my',
  'she',
  'snatch',
  'running',
  'fighting',
  'yelling',
  'it',
  'she',
  'I',
```

Create a Text Generator

We're going to create a function that generates sentences. It will take two things as inputs:

- The dictionary you just created
- The number of words you want generated

Here are some examples of generated sentences:

```
'Shape right turn– I also takes so that she’s got women all know that snail-trail.'
```

```
'Optimum level of early retirement, and be sure all the following Tuesday… because it’s too.'
```

In [5]:
```python
import random

def generate_sentence(chain, count=15):
    '''Input a dictionary in the format of key = current word, value = list of next words
       along with the number of words you would like to see in your generated sentence.'''

    # Capitalize the first word
    word1 = random.choice(list(chain.keys()))
    sentence = word1.capitalize()

    # Generate the second word from the value list. Set the new word as the first word. Repeat.
    for i in range(count-1):
        word2 = random.choice(chain[word1])
        word1 = word2
        sentence += ' ' + word2

    # End it with a period
    sentence += '.'
    return(sentence)
```

In [6]:
```python
generate_sentence(ali_dict)
```

Out[6]:
```
'Since I need to be… this shit turns out, he’s the labor by the umami.'
```

Additional Exercises

1. Try making the generate_sentence function better. Maybe allow it to end with a random punctuation mark or end whenever it gets to a word that already ends with a punctuation mark.

In [ ]:
```python

```