**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.

*# Read in the corpus, including punctuation!*

**import** **pandas** **as** **pd**

data = pd.read\_pickle('corpus.pkl')

data

*# Extract only Ali Wong's text*

ali\_text = data.transcript.loc['ali']

ali\_text[:200]

## 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

**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

*# Create the dictionary for Ali's routine, take a look at it*

ali\_dict = markov\_chain(ali\_text)

ali\_dict

**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.'

**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)

generate\_sentence(ali\_dict)

**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.