Project Writeup and Reflection: Markov text analysis
Nora Mohamed

Project Overview:
For testing, I initially used projectgutenburg.com to get a novel (The Count of Monte Cristo) and test my code using that. However, my goal was to use the titles of articles from BuzzFeed.com. The goal of this project was to randomly generate BuzzFeed article titles, using a Markov analysis. I used dictionaries with list values to generate those titles.

Implementation:
My Markov analysis works by creating dictionaries where words are keys and the values are a list of all the words that immediately follow the keys. For example, the sentence "Hello Bob how is your brother Bob doing?" would generate a dictionary {"Hello": ["Bob"], "Bob": ["how", "doing?"], "how": ["is"], ...et cetera}. The code works by first splitting the lines into a list of words. If the user wants to generate sentences by using two word couplets, then they insert a prefix value of 2. If they want to use word triplets, they use a prefix value of 3, and so on. By default, the prefix value is 1 because I don't have enough BuzzFeed article titles to make new titles that are different with a high prefix value. Using a higher prefix value works well with novels, however. Next, the script will create a dictionary (as described above). Using that dictionary, it creates a sentence (if the user wants, all sentences can start with first words of sentences from the text) that ends at punctuation or at the length the user inputs. To get BuzzFeed article titles, I use Pattern to take the HTML of certain pages and find the titles based on the HTML tags around them (since all titles have similar HTML tags).

One design decision where I had to choose between multiple alternatives was in how I generated my sentences. How do I want the sentences to look like? Should the first word of the sentence I generate be a first word in a sentence from the text I used? How should the sentences be completed? I decided to give the user the option to start with a likely first word in a sentence. Sentences will end if they reach the length that the user inputs, or if the word generated has punctuation that indicates that it's the end of the sentence (e.g., punctuation like: .", ?", !", ., ?, !). I thought that if I ended sentences this way, the sentences would probably make more sense and it wouldn't require the user to manually shorten sentences as much. The downside is that sentences can be one or two words long.

Another design decision I made was the method to create the word groups in the dictionary. Since I wanted to include the ability to make word couplets/triplets/etc., I first tried to create the word groups by splitting the sentence every two/three/etc. words. In the end, I just split the the sentences by words and added words together to form groups instead.

Results:
One problem I have is that I don't have enough BuzzFeed article titles to make a lot of interesting sentences. Due to the format of the website, it's troublesome to do since you'd have to keep looking through pages. Writing this now, I found out that there are BuzzFeed archives that I could've used (which I will implement in the future). Having more article titles would provide more randomness and hopefully create titles that make more sense. I also had a lot of trouble finding a good method to create the word groups, as I described under "Implementation".

I generated pretty interesting BuzzFeed article titles. Some of my favorites (that actually make some sense) include:

> Can You Guess The "Gilmore Girls"
> 19 Peanut Butter Recipes That'll Take Us Back To Netflix, With Murder"

Do You Secondhand Embarrassment
Top Universities Finds Huge Meme
Don't Believe The One Thing About An All-Around Transportation Platform
32 Behind-The-Scenes Photos That Will Restore Your WCW

And my personal favorite:
Would John Travolta Offend You NSFW?

Reflection:
Starting out the project and knowing what I wanted to do went well. I knew what I wanted to do, however I had some problems finding a good method to do what I wanted. For example, when splitting sentences up into words, it took me multiple tries to create a good method for creating word groups. I also spent a lot of time trying to add dictionaries together without making pre-existing keys replace each other (which turned out to be very simple). Testing it was also a bit difficult since the project worked by randomly generating sentences. I learned a lot from this project, although I wish I explored Pattern more than I did (I only used it to grab HTML from BuzzFeed).