Mini Project 3 Report

Project Overview

For this project, I used Markov chaining to emulate the style of Shakespeare's *Hamlet,* the script of *Hamilton: An American Musical,* the two works in combination, and a collection of several Donald Trump speeches.

Because of the nature of Markov chains, I did not expect my end results to be particularly cohesive; that being said, it is much more amusing if the text created has some sort of flow or regard for syntax. In addition, especially in the case where *Hamilton* and *Hamlet* were being used collectively to create a new text, I wanted to include a sort of fail-safe feature that prevents the synthesized text from continuing for too long in a completely determinant nature, thus quoting a passage word-for-word.

Implementation

Because my several programs work in largely the same way, I will describe my implementation in the context of my "Hamleton" synthesis program. After processing both texts to remove things like character names, stage directions, and copyright information, I added the strings for *Hamlet* and *Hamilton* together and then split the resulting string into a list of words.

I chose to keep punctuation attached to words for this program. This allows the resulting synthesized text to have punctuation added organically and account for which words tend to occur only after a punctuation mark. The drawback to this approach is that the sample size of the word choices is effectively reduced, because it treats "Hamilton" as completely independent from "Hamilton," or "Hamilton!"

My program analyzed the list of words to compile a dictionary where the keys are all unique words in the two works and the results are all words that have ever immediately followed the key. In addition, I compiled a separate dictionary that analyzes each pair of consecutive words in the two works keyed to a list of words that have ever immediately followed that unique word pair. To randomly generate text, the program randomly selects an entry from the second dictionary based on the previous two generated words. If no entry exists, or if there has only been a single option for the last five consecutive selections, it instead uses the first dictionary of single words. This ensures that the program does not get into a "determinant loop" and quote word-for-word a long passage from either work that happens to have particularly unique language.

The Donald Trump speech simulator works similarly, but it takes a string as a parameter which it prioritizes over all other options when selecting from a dictionary. For instance, if the previous two words were "I" and "am," the dictionary entry may look like ["Donald", "bigly", "great"], but if "bigly" was passed as a parameter, it will select this entry rather than a random one. In this sense it can simulate themed speeches about China, walls, or emails.

Results

Hamilton:

> 'I am throwing away my shot I lay at your service, sir I just need to write at where we land the revolution’s imminent. What we are free ready to beg, steal, borrow, or so you should have told me wrote to the bride!'
>
> 'News about neutrality with britain and win a victory for the shit now, somebody gotta shovel it! Peach fuzz, and he wrote his first refrain, a legacy? It’s planting seeds in the midst of my love!'

Hamlet:

> "To me: till I had aim'd them. So please you to go directly to the cannoneer without, the cannons to the extent allowable by law. Will be kind: thus bad begins, and peace, that inward breaks, and graveness."
>
> "Where joy meet what I have you any further trade with devotion's visage and pious action we do sugar o'er the house and hurt my behove, o, methought there with fantastic garlands did these bones cost no further."

Hamleton:

> 'Be damned custom have not your speed?! Hamilton! Ha boy! Come, and I knew your father; but that I have heard the way schuyler. It’s only got the tune of the matter.'
>
> 'Fundamental truths to be your foil, Laertes; in mine ignorance your skill shall, like a memory when’s it gonna get shot! Check stubs. From this post-haste and romage in the room… oh oh I would have much to do a thing.'

Donald Trump (seed of “china”):

> "Corrupt pacs is a disaster. It’s not going anywhere anyway, but we’ll see what happened — our leaders, and we talk about china how they’re doing it — she’s already gotten a dose last night, but hopefully it'll all over the years. The fact is that a couple of years, there won’t be a much finer party, a much better for everybody. But china has 18,000 or 13,000 miles, because you know, china comes over and over again, and has emptied our states and has done more than that. And china because they devalued their currency and on and on."

"Corrupt pacs is a disaster. It’s not going anywhere anyway, but we’ll see what happened — our leaders, and we talk about china how they’re doing it — she’s already gotten a dose last night, but hopefully it'll all over the years. The fact is that a couple of years, there won’t be a much finer party, a much better for everybody. But china has 18,000 or 13,000 miles, because you know, china comes over and over again, and has emptied our states and has done more than that. And china because they devalued their currency and on and on.

Shakespearian couplets:

“Tell the ambassador that you might
The noble father much more light!

Slave that be, or i will
Of a winking, mute and still,

Times; and that speech doth move
My heart's core, ay, my love!”

Reflection:

I think that this project was largely successful. I had a very solid idea of what specific functions I needed to complete it effectively, and I followed it with only minor modifications. The only frustrating part was probably cleaning up the source text; keeping punctuation in the list of words meant that I had to catch all cases of “exeunt” or character speech with different combinations of brackets and other punctuation to make sure I caught all cases (and I still have occasional instances of these). In addition, the rhyming function of the couplet program was particularly tough, because I wrote it myself using build-in string operations.

I was surprised at how versatile my resulting code was, because I could apply more or less the same operations to *Hamlet*, *Hamilton*, and Trump speeches with a few minor modifications. This is probably a useful attribute of any program that I could write; to have it able to be used for several applications.

Conclusion:

I felt the need to include an extra section, but I’m too lazy to write it myself. Here is a Markov chain of this report.

Function of the nature of markov chains, I could apply more or less the second dictionary based on the need to include an extra section, but i’m too lazy to be used for several applications.because My program randomly selects an entry rather than a random one. In addition, the rhyming function of my “hamleton” synthesis program. This ensures that the sample size of the program randomly selects an entry from the second dictionary based on the previous two words were “i” and hamlet were being used markov chaining to emulate the same way, I used collectively to create a separate dictionary that analyzes each pair of several donald trump speeches.