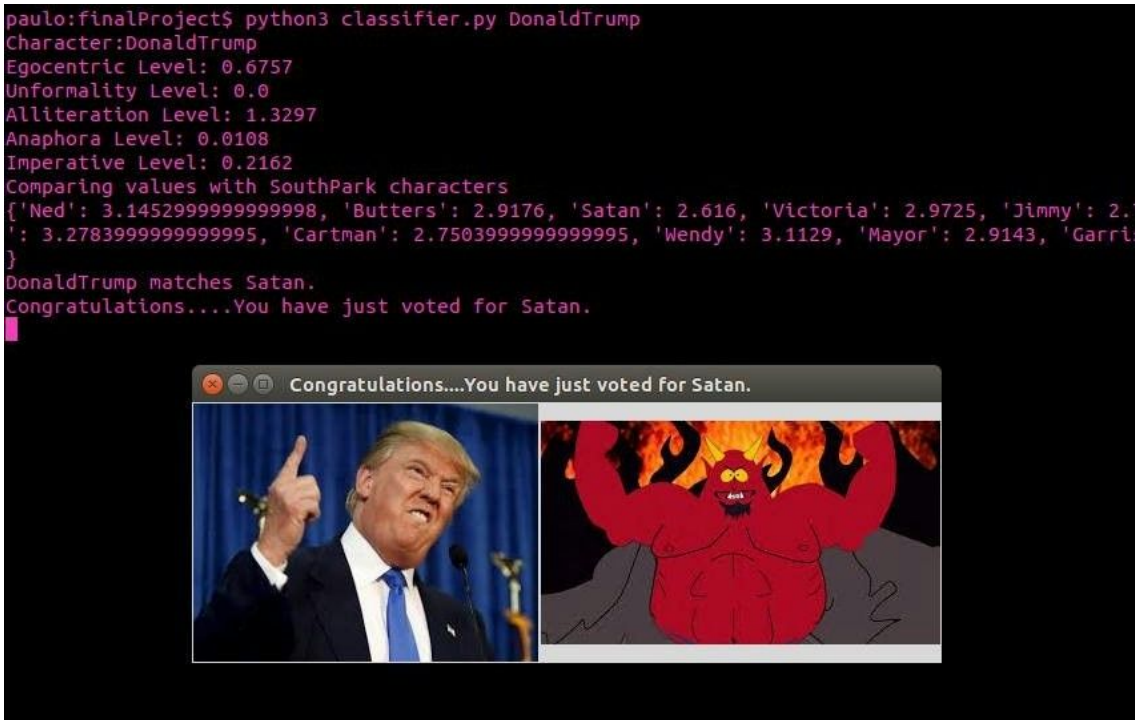
1. **Overview of the project:**

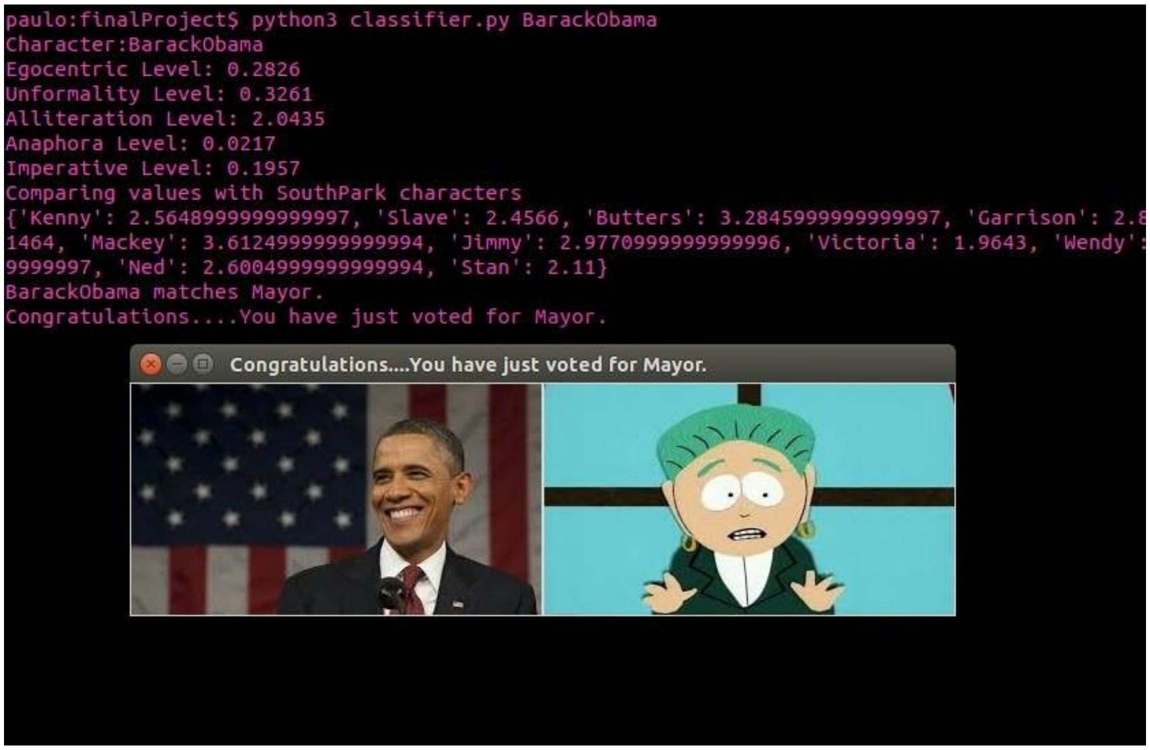
Our project is a *South Park Character Classifier* that matches you with one of the main characters *Cartman*, *Stan* or *Kyle* based on the way you talk! We thought it would be particularly interesting to input the language data of political figures, specifically *Donald Trump, Barack Obama* and *George W. Bush,* and discover which character, based on their speech patterns, they most likely compare to. The classifier analyzes the characters' scripts in contexts of lexical choice; affirmation/negation; egocentrism; formality; alliteration; anaphora; attributing cause; questions and exclamations. Our goal for this project is to showcase how character traits can be translated through linguistic markers.

2. **Results:**

Version 1.0



```
paulo:finalProject$ python3 classifier.py DonaldTrump
Character:DonaldTrump
Egocentric Level: 0.6757
Unformality Level: 0.0
Alliteration Level: 1.3297
Anaphora Level: 0.0108
Imperative Level: 0.2162
Comparing values with SouthPark characters
{'Ned': 3.1452999999999998, 'Butters': 2.9176, 'Satan': 2.616, 'Victoria': 2.9725, 'Jimmy': 2.
': 3.2783999999999995, 'Cartman': 2.7503999999999995, 'Wendy': 3.1129, 'Mayor': 2.9143, 'Garri
}
DonaldTrump matches Satan.
Congratulations....You have just voted for Satan.
```



```
paulo:finalProject$ python3 classifier.py BarackObama
Character:BarackObama
Egocentric Level: 0.2826
Unformality Level: 0.3261
Alliteration Level: 2.0435
Anaphora Level: 0.0217
Imperative Level: 0.1957
Comparing values with SouthPark characters
{'Kenny': 2.5648999999999997, 'Slave': 2.4566, 'Butters': 3.2845999999999997, 'Garrison': 2.8
1464, 'Mackey': 3.6124999999999994, 'Jimmy': 2.9770999999999996, 'Victoria': 1.9643, 'Wendy':
9999997, 'Ned': 2.6004999999999994, 'Stan': 2.11}
BarackObama matches Mayor.
Congratulations....You have just voted for Mayor.
```

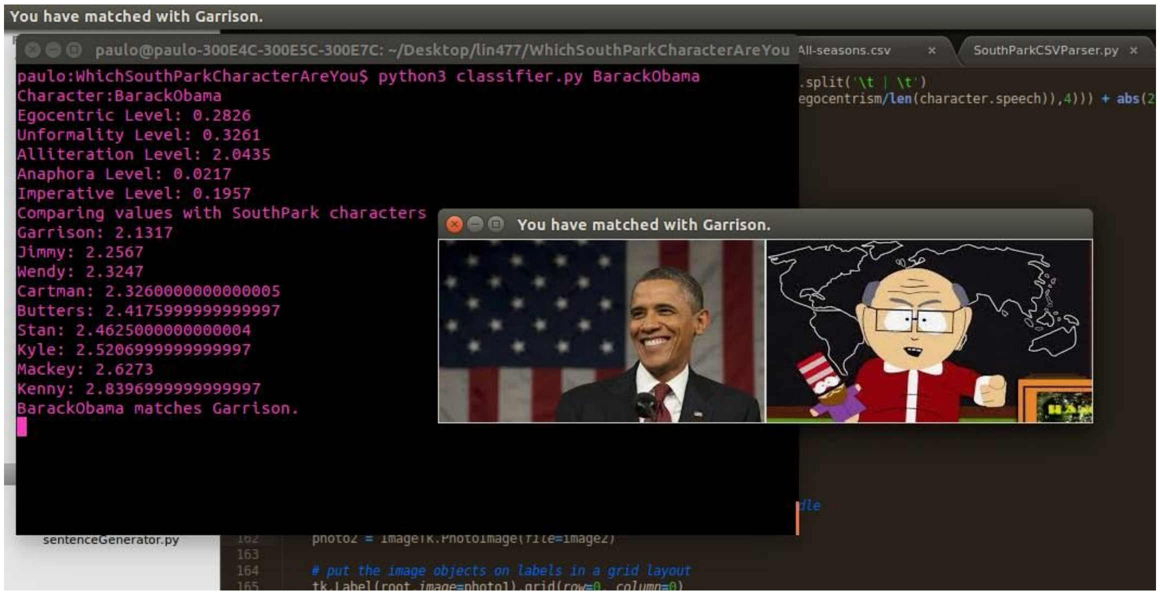
Results for version 1.0:

When we ran the classifier the first time, we got single character match results with less features as seen above.

Results for version 2.0: (code and data for this is located in the overFittedData folder)

When we ran the classifier the second time, we can see how the Presidents who for the most part the general public view less positively, are more closely related to the South Park Characters (can be seen by the lower numbers) than the Presidents who are viewed more positively. Heuristic used in determining the below numbers is the total manhattan distance between each attribute where certain attributes are multiplied by some constant because we suspect them to contribute more in the classifying of such characters. (Refer to results below)

	Egocentric Level	Unformality	Alliteration Level	Anophora Level	Imperative level
Obama	-0.6304	0.1522	2.0435	0.0217	0.1304
Bush	-0.24	0.1333	1.4067	0.0	0.0733
Trump	0.0649	0.0	1.3297	0.0108	0.2054
Sanders	-0.6757	0.027	3.7297	0.027	0.1351

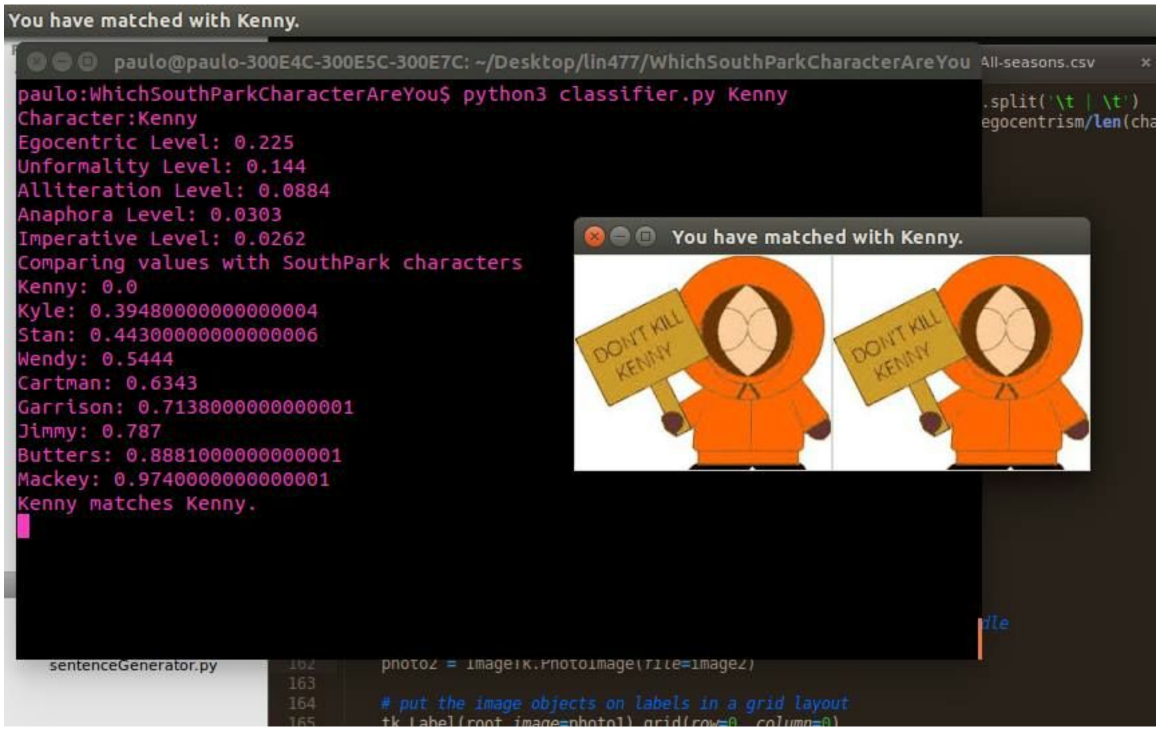


You have matched with Garrison.

paulo@paulo-300E4C-300E5C-300E7C: ~/Desktop/lin477/WhichSouthParkCharacterAreYou

```
paulo:WhichSouthParkCharacterAreYou$ python3 classifier.py BarackObama
Character:BarackObama
Egocentric Level: 0.2826
Unformality Level: 0.3261
Alliteration Level: 2.0435
Anaphora Level: 0.0217
Imperative Level: 0.1957
Comparing values with SouthPark characters
Garrison: 2.1317
Jimmy: 2.2567
Wendy: 2.3247
Cartman: 2.3260000000000005
Butters: 2.4175999999999997
Stan: 2.4625000000000004
Kyle: 2.5206999999999997
Mackey: 2.6273
Kenny: 2.8396999999999997
BarackObama matches Garrison.
```

```
.split('\t | \t')
egocentrism/len(character.speech)),4))) + abs(2
photo2 = ImageTk.PhotoImage(file=image2)

# put the image objects on labels in a grid layout
tk.Label(root,image=photo1).grid(row=0, column=0)
```

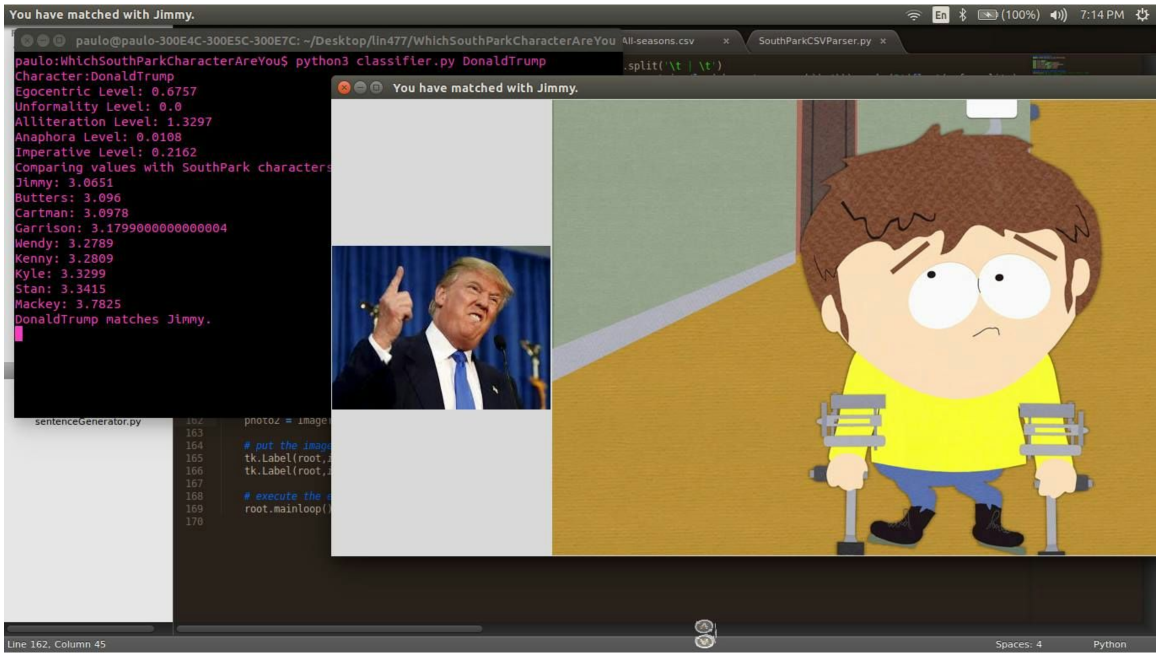


You have matched with Kenny.

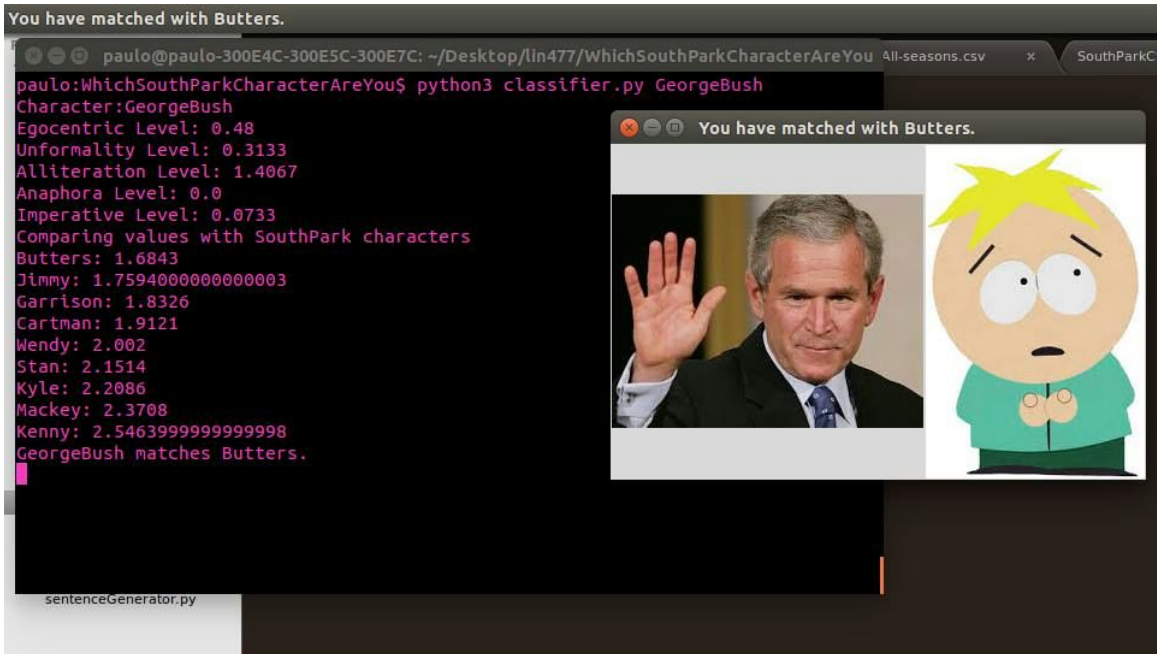
paulo@paulo-300E4C-300E5C-300E7C: ~/Desktop/lin477/WhichSouthParkCharacterAreYou

```
paulo:WhichSouthParkCharacterAreYou$ python3 classifier.py Kenny
Character:Kenny
Egocentric Level: 0.225
Unformality Level: 0.144
Alliteration Level: 0.0884
Anaphora Level: 0.0303
Imperative Level: 0.0262
Comparing values with SouthPark characters
Kenny: 0.0
Kyle: 0.39480000000000004
Stan: 0.44300000000000006
Wendy: 0.5444
Cartman: 0.6343
Garrison: 0.7138000000000001
Jimmy: 0.787
Butters: 0.8881000000000001
Mackey: 0.9740000000000001
Kenny matches Kenny.
```

```
.split('\t | \t')
egocentrism/len(cha
photo2 = ImageTk.PhotoImage(file=image2)

# put the image objects on labels in a grid layout
tk.Label(root,image=photo1).grid(row=0, column=0)
```


You have matched with Jimmy.
paulo@paulo-300E4C-300E5C-300E7C: ~/Desktop/lin477/WhichSouthParkCharacterAreYou
All-seasons.csv
SouthParkCSVParser.py
sentenceGenerator.py
Line 162, Column 45
Spaces: 4
Python

```
paulo:WhichSouthParkCharacterAreYou$ python3 classifier.py DonaldTrump
Character:DonaldTrump
Egocentric Level: 0.6757
Unformality Level: 0.0
Alliteration Level: 1.3297
Anaphora Level: 0.0108
Imperative Level: 0.2162
Comparing values with SouthPark characters
Jimmy: 3.0651
Butters: 3.096
Cartman: 3.0978
Garrison: 3.1799000000000004
Wendy: 3.2789
Kenny: 3.2809
Kyle: 3.3299
Stan: 3.3415
Mackey: 3.7825
DonaldTrump matches Jimmy.
```


You have matched with Butters.
paulo@paulo-300E4C-300E5C-300E7C: ~/Desktop/lin477/WhichSouthParkCharacterAreYou
All-seasons.csv
sentenceGenerator.py

```
paulo:WhichSouthParkCharacterAreYou$ python3 classifier.py GeorgeBush
Character:GeorgeBush
Egocentric Level: 0.48
Unformality Level: 0.3133
Alliteration Level: 1.4067
Anaphora Level: 0.0
Imperative Level: 0.0733
Comparing values with SouthPark characters
Butters: 1.6843
Jimmy: 1.7594000000000003
Garrison: 1.8326
Cartman: 1.9121
Wendy: 2.002
Stan: 2.1514
Kyle: 2.2086
Mackey: 2.3708
Kenny: 2.5463999999999998
GeorgeBush matches Butters.
```

Obama matched with Jimmy with a manhattan distance of 11.83375
Sanders matched with Jimmy with a manhattan distance of 16.54445
Trump matched with Jimmy with a manhattan distance of 6.38515
Bush matched with Jimmy with a manhattan distance of 6.84455

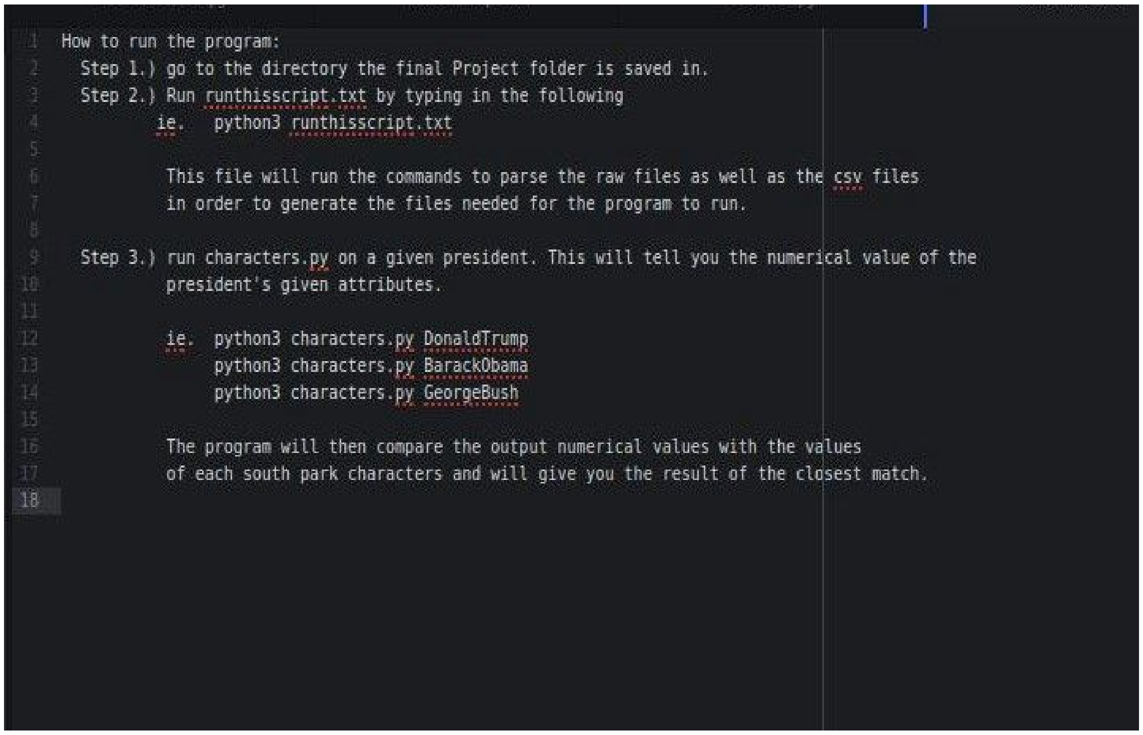
Sample result for the sentence generator

```
$ python3 sentenceGenerator.py Wendy 5

But Lisa, that 's great! Have you told Butters that you like him? Lisa, this is
exactly what you need! Take Butters to see a movie or something! It 'll do
wonders for your confidence! What the fuck is your problem?! Breast cancer is n't
funny! Breast cancer is killing people! This is a serious issue, Eric! What you 're
doing is very offensive! Will somebody do something? Every week he gets worse
and nobody does anything! No you are n't! What is your problem?! Are you just
an asshole?! Is that it?! Yeah! Lisa Berger asked you out and you called her fat?!
Do you have any idea how you made her feel?! She 's a really nice girl! She 's a
little overweight, but that 's pretty normal for a girl in the fourth grade! What?!
This is a fantasy, you moron! You ever heard of Photoshop?! Kim Kardashian is
a short, overweight woman who manipulates her image and makes average girls
feel horrible about themselves! Look it up, stupid! In real life, Kim Kardashian has
the body of a hobbit! You 're gon na be in real trouble when the teachers find out
what you said to that poor girl! Oh good.

$ python3 sentenceGenerator.py Wendy 5
I made that egg.
```

This shows the randomness of the sentence generator

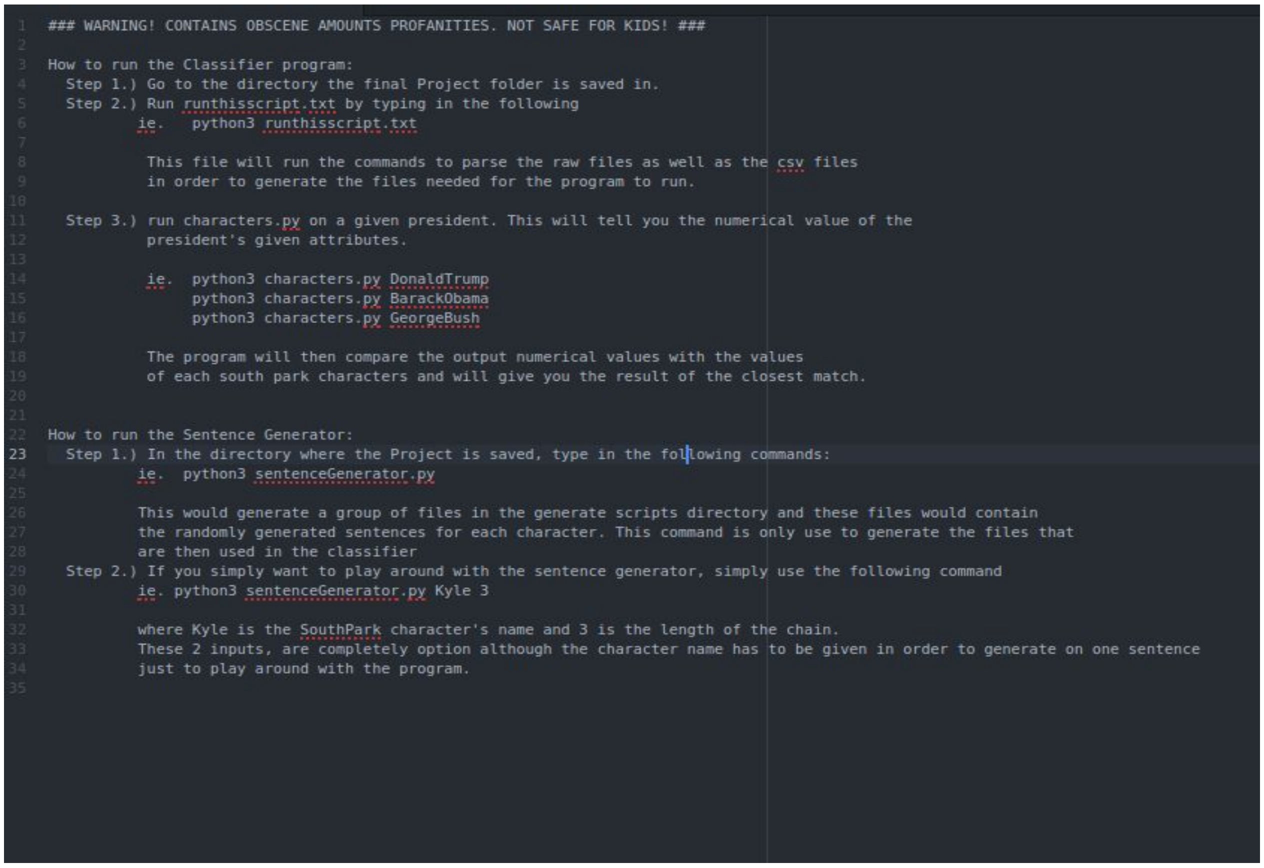
3. Running:

Version 1.0



```
How to run the program:
  Step 1.) go to the directory the final Project folder is saved in.
  Step 2.) Run runthisscript.txt by typing in the following
          ie.   python3 runthisscript.txt

           This file will run the commands to parse the raw files as well as the csv files
           in order to generate the files needed for the program to run.

  Step 3.) run characters.py on a given president. This will tell you the numerical value of the
           president's given attributes.

           ie.  python3 characters.py DonaldTrump
                python3 characters.py BarackObama
                python3 characters.py GeorgeBush

           The program will then compare the output numerical values with the values
           of each south park characters and will give you the result of the closest match.
```

Version 2.0



```
### WARNING! CONTAINS OBSCENE AMOUNTS PROFANITIES. NOT SAFE FOR KIDS! ###

How to run the Classifier program:
  Step 1.) Go to the directory the final Project folder is saved in.
  Step 2.) Run runthisscript.txt by typing in the following
          ie.   python3 runthisscript.txt

           This file will run the commands to parse the raw files as well as the csv files
           in order to generate the files needed for the program to run.

  Step 3.) run characters.py on a given president. This will tell you the numerical value of the
           president's given attributes.

           ie.  python3 characters.py DonaldTrump
                python3 characters.py BarackObama
                python3 characters.py GeorgeBush

           The program will then compare the output numerical values with the values
           of each south park characters and will give you the result of the closest match.


How to run the Sentence Generator:
  Step 1.) In the directory where the Project is saved, type in the following commands:
          ie.  python3 sentenceGenerator.py

          This would generate a group of files in the generate scripts directory and these files would contain
          the randomly generated sentences for each character. This command is only use to generate the files that
          are then used in the classifier
  Step 2.) If you simply want to play around with the sentence generator, simply use the following command
          ie. python3 sentenceGenerator.py Kyle 3

          where Kyle is the SouthPark character's name and 3 is the length of the chain.
          These 2 inputs, are completely option although the character name has to be given in order to generate on one sentence
          just to play around with the program.
```

4. **How it works:**

It should be noted that linguistic features from South Park characters were of interest and not the linguistic features of the political figures. We searched the scripts for general markers that would be significant in the speech of the characters. However, we found that by searching for linguistic characteristics in political figures, we had a base for more things to look for in the characters' scripts, for example, alliteration and use of contractions. We verified that the frequencies of these features were significant enough in the characters' speech to include it in the classifier.

- Helper Scripts:
  - SouthParkCSVParser.py
    - This script goes through the entire csv file containing all the lines of every character in South Park and organizing then into their own character file.The character files are only of the main characters that have more than around 1300 lines.There is also a file called allChars.txt containing all the maincharacter's lines

  - PresidentialSpeechParser.py
    - This script goes through a given President's Speech and takes out all the HTMLnoise if any(since it was taken online) and writes each sentence in one line for easier readability
  - characterAttr.py
    - This script calculates the numerical values of each character and writes them to a filecalled AllCharAttributes.txt. This file will be used later by the classifier.

- Main Files:
  - classifier.py
    - This file calculates the numerical attributes of a given Person provided the file has access to afile containing the person's speech/ group of texts. The classifier quantifies the attributes found in the file and provides a match between the Person and a South Park character.

  - sentenceGenerator.py

- This file uses the Markov chain model trained on a given character's speech to construct a sentence.It uses no linguistic features beacuse the whole point of this program is to point out that without thelinguistic markers used in the classifier, construct a sentence using the Markov model causes the sentence to lose its identity.We prove this by classifying a group of text generated from the Markov model and seeing that the generated text will be classified as a different character.

Parameters/ Things for the program to look out for:

	How to identify	Focus	Examples/variations	Suggested method
Syntax http://www.winwaed.com/blog/2011/11/08/part-of-speech-tags/	Simple vs complex	EX/ NN + VP + DT + (JJ) + NP Anything different		
http://www.winwaed.com/blog/2011/11/08/part-of-speech-tags/	Imperative	VB + IN + NN/NNP VB + PRP HAVE + TO	Commands Use of the 2 person *Look at Paris.* *Do it* Marked by the sentence structure beginning with: VERB + PREPOSITION “HAVE TO” constructions ie *We have to*	Pos_tag contains posX

				contains *have* + *to*
Negative vs Affirmative	negative markers positive markers	no, not yes, yeah	“no” ‘It’s not!’ ‘Noooo” “n’t” “yes” “Yehehehesss” “Yehhhs”	Lemmatize the variations of each Contains word
Egocentrism	Personal pronouns	I vs we	‘I’, ‘me’, ‘my’ ‘we’, ‘us’, ‘our’	Contains word
Formality	Contractions	Apostrophes	“I’m” “Who’s” “We’re” “Should’ve” “Don’t”	Contains character ‘
Alliteration	initial letter repetition	Initial letters	Big brown bouncing bugs	Contains 2+ words where Initial non-alpha is same character
Anaphora	Emphasis by word repetition	Repeated words	‘I really really wanna go to Casa Bonita’	Contains same word x2+

Political Figures

Political Figure	**Linguistic Marker**	**Example**
Bush	**Contractions**	Because Bush sounded “more relaxed,” he was perceived as having been despite his obvious connections — less shaped by

		Washington. Language use reinforced that view. Gore frequently did not use contracted forms such as “I’m” or “It’s,” whereas candidate Bush almost always did. This tiny, almost imperceptible difference in their speaking styles as noted by linguist Geoffrey Nunberg → http://www.pbs.org/speak/seatosea/standardamerican/presidential/
Trump http://digg.com/video/donald-trump-linguistics-answer-question	**Repetition**	His rhetoric, and more specifically his speaking style, is another. Trump, as Fix contributor Barton Swaim wrote a while back, has a habit of punctuating each argument with short, declarative sentences. And then he often repeats them. → https://www.washingtonpost.com/news/the-fix/wp/2016/01/15/this-video-of-donald-trump-repeating-himself-is-pretty-great-video/
	Imperative	Commands Use of the 2 person *Look at Paris. Look at what happened in Paris.* Marked by the sentence structure beginning with: VP (PP) ALSO marked by containing “HAVE TO” constructions ie *We have to* Trump often uses short, imperative sentences, Hayes says. And according to a recent linguistic study, his speech rests at about a fourth-grade level. → http://www.techinsider.io/how-donald-trump-talks-2016-3
	Contractions	A **contraction** is a shortened version of the written and spoken forms of a word or syllable created by omission of internal letters and sounds.

		I am → I'm Did not → Didn't
	Monosyllabic words (78%) **Frequency of bi-syllabic words (17%)** **Almost never more than 2 syllables**	???
Obama	**Repetition**	The repetition of a word or phrase at the start of successive clauses or sentences. Usage of repetition as a rhetorical device helps to convey and reinforce a certain message in a successive manner that resonates with the audience like layering a Lego brick atop another sequentially. If you listen to Obama's speeches enough, he uses them generously. I.e → "This country has more wealth than any nation, *but that's not what* makes us rich. We have the most powerful military in history, *but that's not what* makes us strong. Our university, our culture are all the envy of the world, *but that's not what* keeps the world coming to our shores."
	Alliteration	The occurrence of the same letter or sound at the beginning of adjacent or closely connected words. I.e → "Through *it all, we have never relinquished our skepticism of central authority, nor have we succumbed to the*

		fiction that all society's ills can be cured through government alone. Our celebration of initiative and enterprise; our insistence on hard work and personal responsibility, are ***constants*** *in our* ***character****."*
	The use of first person plural pronouns	*We* instead of *I*

5. **Project Lessons:**

Some things we tried to implement in Version 2.0, but caused problems for our classifier were implementing features that looked at affirmation and negation, Part-of-Speech tagging, frequency of questions, and including characters with few lines.

Affirmation and negation in a character's script and frequency of questions were problematic for the classifier because it seemed that these features occurred more in a single character than to be distributed across them. This caused the classifier to match the input, political figures' scripts, to only one character although the degree of how similar they are to that one character varied by some factor. In sum, this was a case of overfitting the classifier.

Implementing Part-of-Speech Tagging was also an issue. Through the script analysis, we were able to determine that the script wasn't structured the same way as the sentences the POS tagger was trained on which is why it yielded inconsistencies with the tagging. This issue is likely due to the script being written as discourse, which strays from the rigid structures of formal written language.

Finally, including South Park characters with few lines in comparison to others in the classifier caused a lot of problems. Since we're giving the characters a percentage based numerical set of attributes, recurring characters who only have a few lines would have a larger percentage of a certain attribute For example, a character with only 10 lines who uses particular egocentric linguistic markers and 5 of their lines is considered to be more egocentric than a character with 10000 lines who uses egocentric linguistic markers in 3000 of those lines.

Version 1.0 of the classifier worked successfully as intended because it was not overfitted with features and also, using it on characters with at least 1300 lines proved it to be more reliable than using it on characters with only a few lines but have a higher density of certain attributes.

The Markov model sentence generator trained on a given character works fine with generating random sentences that presumably sounds like the character. This sentence generator was done without any linguistic features but simply using statistics to connect words to form a sentence.Also, we used a group of sentences from from this Markov model and we tried to use the classifier to classify these text to the same character we trained the Markov model on. The result of this was that the generated sentences had lost the attributes that identify the characters causing the classifier identify the next text as a different character. This shows computer science alone can only take us so far when working with natural language processing. Even for sentence generation, it would be better if we had incorporated the linguistic features we have found into the sentence generator so that that Markov Model would be able to pick out the next word of the sentence without losing the distinguishing factor of the character in the sentence.

Overall, version 2.0 was not a success due to overfitting the data. Version 1.0 is different than version 2.0 because it was more of a success due to less over fitted data. In Version 2.0 we added a sentence generator that essentially generates sentences based on the speech of a given character.

Looking at the outcome and results at version 1.0 and version 2.0, we would have to find better fitted features and focus more on word mining of the characters. Also, we would find a way to get around the fact that the NLTK POS tagger focuses on standard syntax, whereas the South Park script is more complex in that it does not follow the regular syntactical pattern of a simple english sentence.

6. References:

1. A Linguistic Analysis Of Donald Trump Shows Why People Like Him So Much. (n.d.). Retrieved April 06, 2016, from http://digg.com/video/donald-trump-linguistics-answer-question

2. (n.d.). Retrieved April 06, 2016, from http://www.pbs.org/speak/seatosea/standardamerican/presidential/

3. 3 things a MIT scientist learned about Trump by studying his debates. (n.d.). Retrieved April 06, 2016, from http://www.techinsider.io/how-donald-trump-talks-2016-3

4. Maraniss, D., & Samuels, R. (n.d.). We noticed Donald Trump says a lot of things twice. He says a lot of things twice. [Video]. Retrieved April 06, 2016, from https://www.washingtonpost.com/news/the-fix/wp/2016/01/15/this-video-of-donald-trump-repeating-himself-is-pretty-great-video/

5. 8 Powerful Speech Techniques that President Barack Obama used to “Wow” the World in his Presidential Victory Speech 2012. (2012). Retrieved April 06, 2016, from http://www.benjaminloh.sg/2012/11/07/8-powerful-speech-techniques-that-president-barack-obama-used-to-wow-the-world-in-his-presidential-victory-speech-2012/

6. What Makes Obama a Good Speaker? (2008). Retrieved April 06, 2016, from http://observer.com/2008/02/what-makes-obama-a-good-speaker/

7. (n.d.). Retrieved April 06, 2016, from http://www.pbs.org/speak/seatosea/standardamerican/presidential/

8. The Raw Story | Bush's problem with, uh, public speaking. (n.d.). Retrieved April 06, 2016, from http://www.rawstory.com/exclusives/droms/bush_public_speaking.html

9. A Revealing Analysis Of Word Frequency On South Park. (n.d.). Retrieved April 06, 2016, from http://www.kotaku.com.au/2016/02/a-revealing-analysis-of-word-frequency-on-south-park/