NLP with South Park

Milestone Report

Taylor Willingham
September 7, 2019

Overview

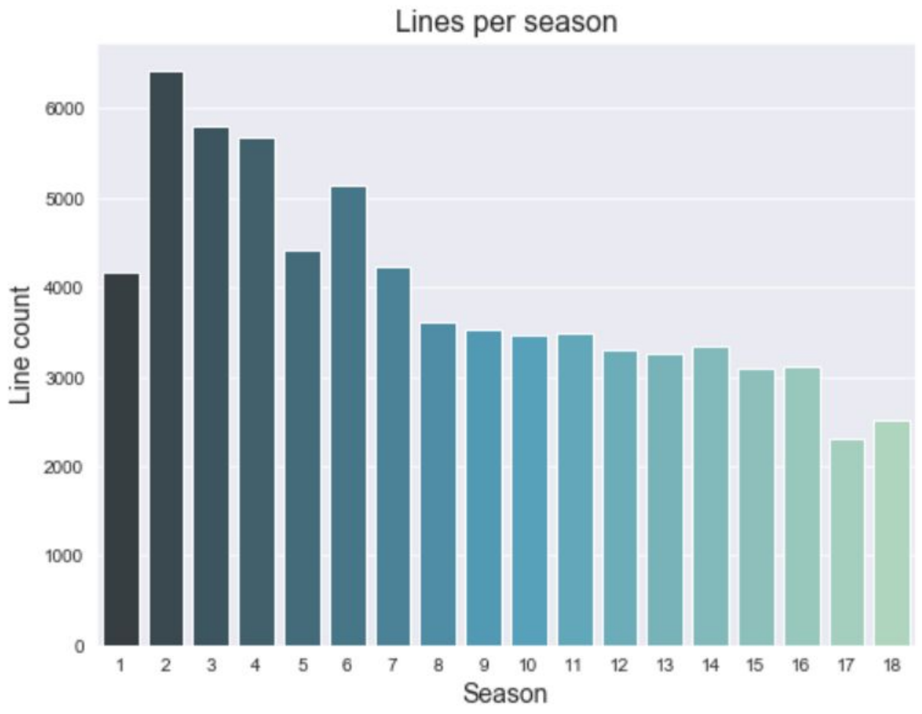
The general intent for this project is to dive into the methods and techniques of natural language processing to get some experience using machine learning on textual data. In order to do so, I have chosen a data set of scripts broken down line by line from episodes of the television show South Park. With this data, my aim is to train a model that, when given a line of dialogue, will be able to accurately predict which character is speaking.

Now, let's be clear: this is more of a fun passion project versus anything substantial. Knowing which South Park character is speaking probably isn't very important, and if you're hearing the line for the first time you can probably already see who is delivering the line. However, the techniques used here have infinite potential applications. It boils down to a classification problem, where labels are predicted based on text. This could be used to sort types of documents, categorize articles by subject, seek out examples of plagiarism, and countless other things. Not to equate South Park with Shakespeare, but it's not all that different from scholars trying to determine whether or not an anonymous piece of literature is from the hand of the great poet and playwright. So, even though this project focuses on a comedy TV show, the potential implications reach much further.

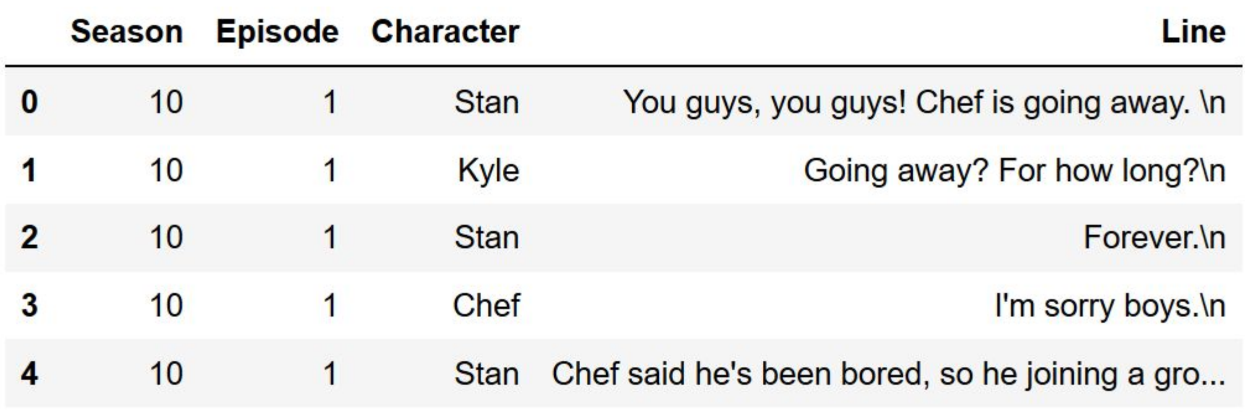
Once the project is finished, I plan to have a jupyter notebook containing all of the code and exposition, a final report and a slide deck.

The Data

As for the data used for this project, it can be found here. Each line of dialogue is treated as a separate observation. Fortunately, South Park is a long running show, so there are quite a few episodes to source from, and this data set contains dialogue covering the first 18 seasons. Here's a rough idea of how many lines, or documents for the purpose of this project, there are within the unaltered data:



There are only two columns that really matter: the character column, which is the label of interest, and the line column, which contains the actual dialogue. In addition to these, there are two other columns for the episode number and the season number for when the line occurred.



	Season	Episode	Character	Line
0	10	1	Stan	You guys, you guys! Chef is going away. \n
1	10	1	Kyle	Going away? For how long?\n
2	10	1	Stan	Forever.\n
3	10	1	Chef	I'm sorry boys.\n
4	10	1	Stan	Chef said he's been bored, so he joining a gro...

Data Wrangling

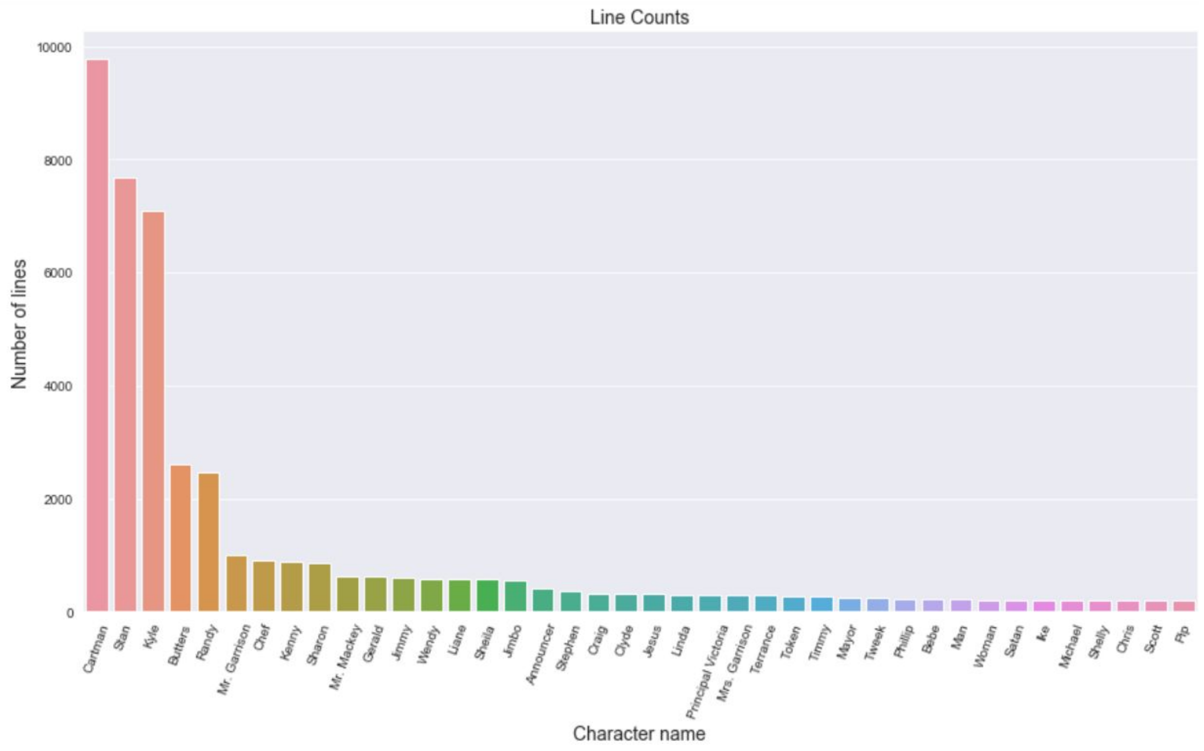
Because algorithms only work on numeric data, I will eventually have to convert the dialogue into numerical vectors in order to solve the problem. With the converted vectors, I can try different tactics using bag-of-words, n-grams, tf-idf and others to find the best approach.

However, I have not had to worry about these steps quite yet. So far, the primary concern has revolved around truncating the data set to focus on the primary characters, and exploring the resulting word corpus.

The first issue to address was a small case of bad data. Apparently, each season had its own subheading, which resulted in 17 rows of extraneous information. With these removed, the next area of focus was determining how to handle all the characters present in the data. Like just about any TV show, South Park uses a lot of guest characters and small supporting characters. Across 18 seasons, this means that there were roughly 4,000 unique characters represented in the data set. This creates a problem for classification. It’s nearly impossible to accurately classify a character that might only have 1-2 lines of dialogue so, in truth, these characters just introduce noise.

There are two options to deal with this issue: group characters together to create new labels, likely a binary classification scenario based on main character or supporting character; or just drop all the supporting characters and simply focus on the primary characters. I chose the latter option, dropping all of the less significant characters and narrowing the focus to the primary ones.

This graph demonstrates who those characters might be:



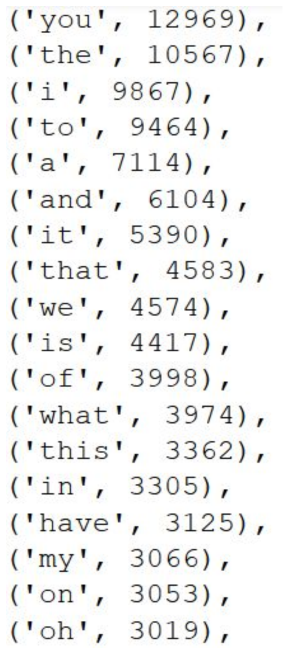
Based on the bars, there appears to be five characters that clearly stand out from the rest. It might be difficult to read those from the image, but they are: Cartman, Stan, Kyle, Butters, and Randy. One caveat, with this approach there will be a significant loss of data. To alleviate this somewhat, I decided to keep some of the main supporting characters, focusing on important authority figures like parents and teachers, grouping them under one label of 'Support Character'. The full list of grouped characters is: Chef, Sharon, Gerald, Liane, Sheila, Stephen, Mr. Garrison, Ms. Garrison, Mrs. Garrison, and Mr. Mackey.

With the data truncated, a total of 35,454 lines remain, and this project is now a multiclass classification problem with a total of six classes.

Word Corpups, Stop Words, and Initial EDA

The next step is to create the word corpus, which eventually will be used to create the word count vector to compare documents numerically. It also allows exploration of the data on a granular level by examining the individual words. After compiling each line of dialogue into one list and removing the new line figure (\n) from the end of each line, we can gain insights on the documents themselves without worrying about the other variables. For example, the average document length is 11.16 words long, reflecting the brief nature of character speech.

To break it down even further, I created an ordered dictionary of word counts to get a dictionary of ranked word frequencies. The most common words with their respective counts are as follows:

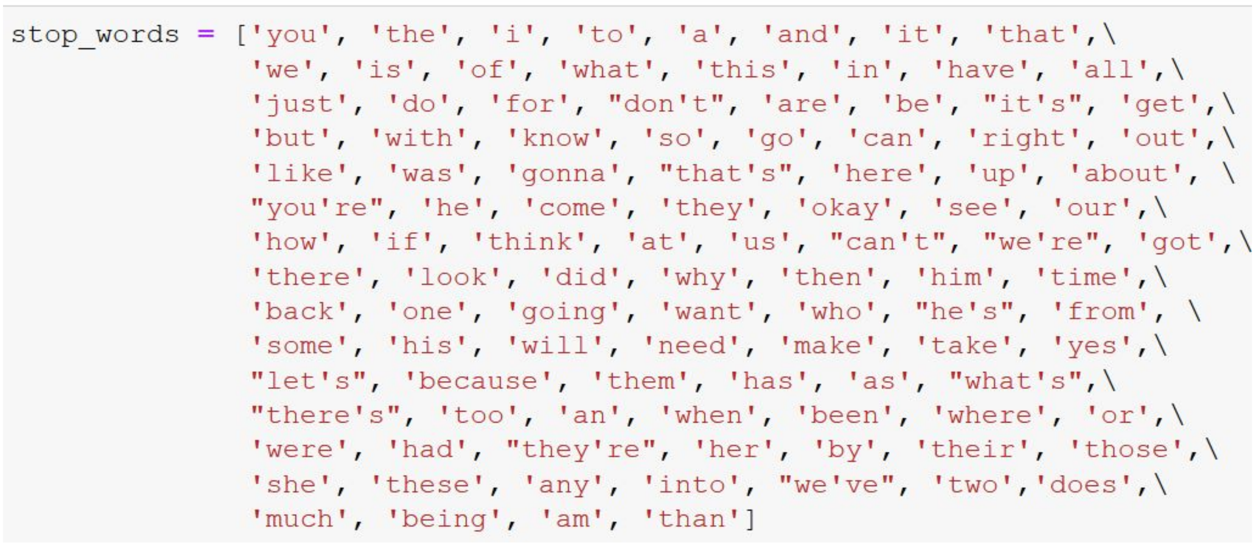


```
('you', 12969),
('the', 10567),
('i', 9867),
('to', 9464),
('a', 7114),
('and', 6104),
('it', 5390),
('that', 4583),
('we', 4574),
('is', 4417),
('of', 3998),
('what', 3974),
('this', 3362),
('in', 3305),
('have', 3125),
('my', 3066),
('on', 3053),
('oh', 3019),
```

This list of words gives us a list of potential stop words, or common that don't provide much information due to their lack of distinctiveness. Words that are commonly used by all characters do not provide much information, so it helps to remove them. On the other hand, some words appear frequently overall, but may not be used the same amount by all characters. Any such words could still be useful for classification. One example is the word 'dude', which has a high overall frequency in the corpus, but is more often associated with the three main characters as opposed to the others. So even though it has a similar frequency to the word 'but', it is useful to leave 'dude' in the corpus while dropping 'but' as a stop word.

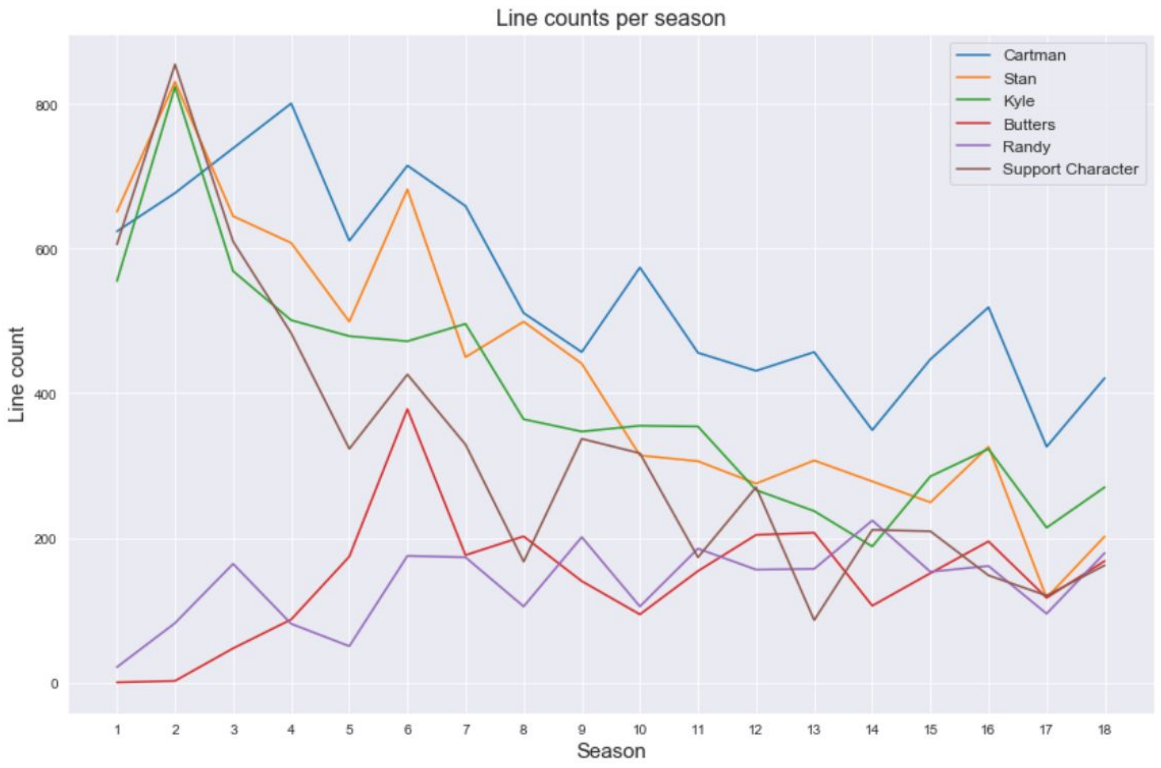
Because overall frequency doesn't tell the full story, I decided to manually check words individually in order to decide which words to drop. I created a function for checking frequencies across the different labels, and using that metric to help make a determination. After going through the first couple hundred or so of the most common words, I narrowed the list down to 106 stop words to discard. Besides ‘dude’, I also chose to keep common words like ‘my’ and ‘mine’ because they had imbalanced distributions, likely due Cartman’s selfish behaviour. One final note: I also chose not to remove ‘no’ and ‘not’ because negation words like those could be useful later when moving beyond a simple bag of words model.

Here’s the final list:



```
stop_words = ['you', 'the', 'i', 'to', 'a', 'and', 'it', 'that',\
               'we', 'is', 'of', 'what', 'this', 'in', 'have', 'all',\
               'just', 'do', 'for', "don't", 'are', 'be', "it's", 'get',\
               'but', 'with', 'know', 'so', 'go', 'can', 'right', 'out',\
               'like', 'was', 'gonna', "that's", 'here', 'up', 'about', \
               "you're", 'he', 'come', 'they', 'okay', 'see', 'our',\
               'how', 'if', 'think', 'at', 'us', "can't", "we're", 'got',\
               'there', 'look', 'did', 'why', 'then', 'him', 'time',\
               'back', 'one', 'going', 'want', 'who', "he's", 'from', \
               'some', 'his', 'will', 'need', 'make', 'take', 'yes',\
               "let's", 'because', 'them', 'has', 'as', "what's",\
               "there's", 'too', 'an', 'when', 'been', 'where', 'or',\
               'were', 'had', "they're", 'her', 'by', 'their', 'those',\
               'she', 'these', 'any', 'into', "we've", 'two','does',\
               'much', 'being', 'am', 'than']
```

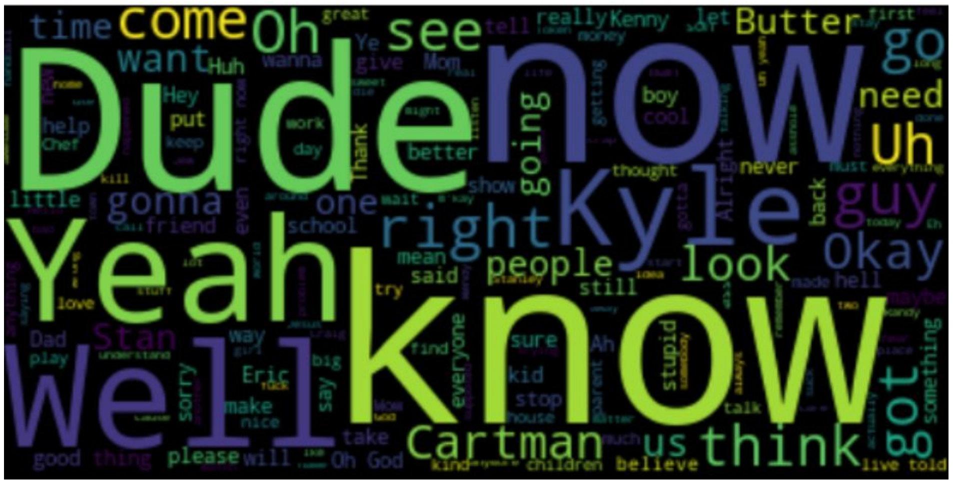
One final thing I also wanted to look at was how the number of lines per character increased or decreased as the seasons progressed. For example, I know that while Butters is one of the most featured characters, he wasn’t part of the show initially. It’s mostly out of curiosity, but if certain characters change in prominence, the use of the words most associated with those characters would likely also change in frequency.



Visual EDA with Word Clouds

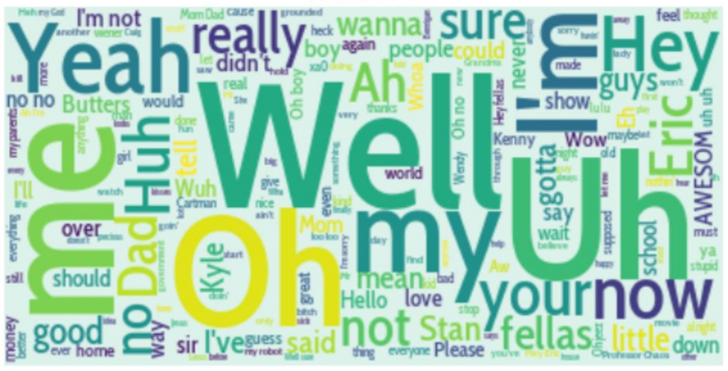
With the word corpus set up and the stop words removed, one way to visualize potential differences in character vocabulary is with word clouds. These will display common words, with the most frequent ones featured more prominently.

Here's a basic example for the entire corpus:

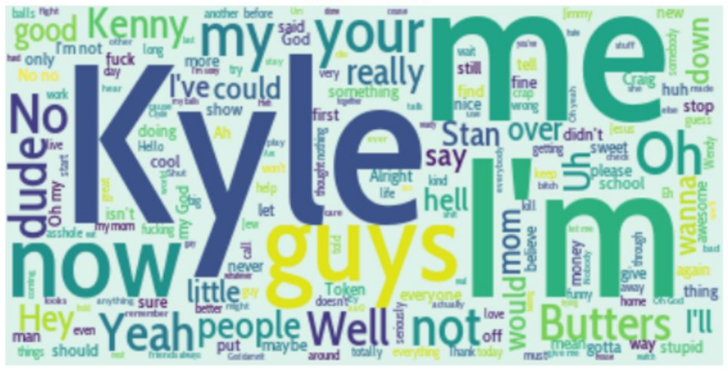


Of more use is to compare word clouds across the different labels to get a sense of how they might differ. After removing the specific stop words from above and cleaning it up a bit, here are the different character word clouds.

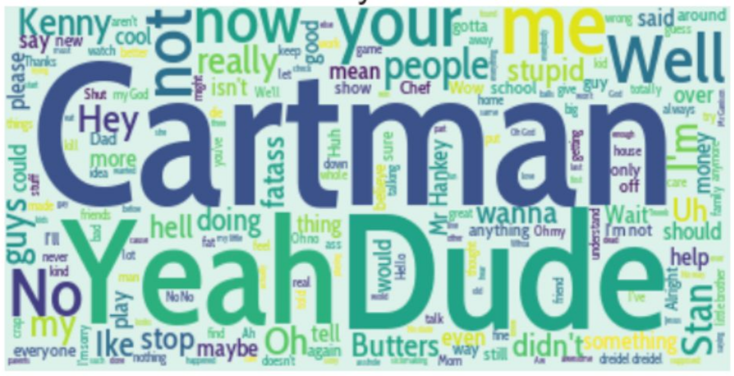
Butters



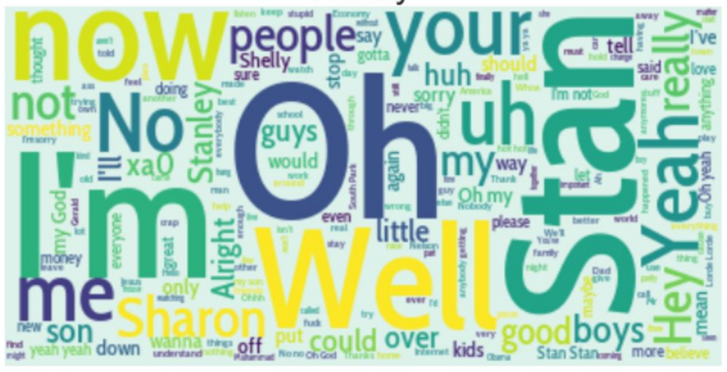
Cartman



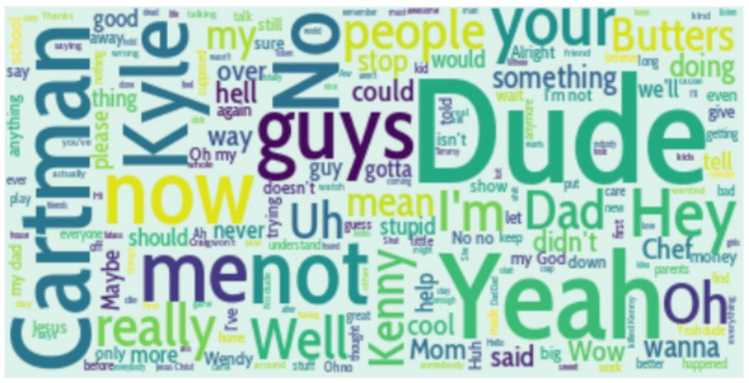
Kyle



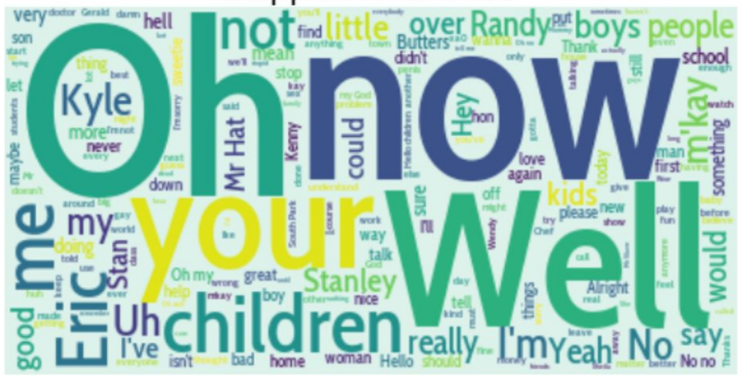
Randy



Stan



Support Character



As we can see there are some differences. Besides the specific words, it’s also interesting to see how some characters like Cartman and Kyle have a few words that really stand out, while another character like Stan has a more even distribution.

Hopefully these distinctions will be strong enough to lead to an accurate model. The next step is to use the word vector and test different algorithms to try and find the best model.