**Report for Homework Zero - Normalization – Connor Homayouni**

this report will also act as the readme file that is posted on my GitHub

introduction to the data selected:

To make the homework more interesting I decided to process and analyze the scripts from the OG Star Wars Trilogy. Specifically the scripts and not just a book that was explaining the theatrical version of the story. I chose a script because there are a bunch of script specific characters that would need to be taken into account when doing normalization and there is also the addition of extra text that wouldn't have been there and a book to supply more context to the reader. Overall it's a more diverse data set or at least that's the logic I used in my mind.

How to use the script(Methodology):

To use my program to analyze this scripts from the first Star Wars trilogy you simply have to download the program homework\_0.py and the text file “the\_original\_trilogy.txt” and make sure that they are added to the same directory. I've built the script so that as long as the text file you want to analyze is in the same directory as the program, you only have to type in the name .txt file and the program will fill in the rest of the path automatically.

There are several options that are available to use for analyzing the text:

"-l", "--lower", help="Convert all words to lowercase."

"-s", "--stem", help="Stem words."

"-sw", "--stop\_word", help="Remove all stopwords from the final."

"-ph",'--search\_phrase', help="Search for a specific phrase in the text."

"-ch", "--find\_characters", help="Search for the total number of times the main characters are mentioned."

The first three options simply do what was required in the homework(lowercasing, stemming, and stop word removal) the final two options are customed to the Star Wars script although they could be adapted to fit other bodies of text. The search phrase option allows the user to type a string that they would like to search for within the text and it returns a graph that shows how many times that exact string appears. This option can also be modified with the first three options, though they will affect the amount of times a phrase appears.

The last option was a fun one. I noticed that many of the most common words in the script word names of the characters. This makes, but I thought it might be nice to be able to separate that data from the overall pool. The final option we'll return a graph that has the amount of times character names appear within the script and remove them from the overall pool of data. This list is specific to Star Wars episode 4, but you can easily go into the code and modify it to have a different list of names to exclude.

As long as the text you would like to analyze is in the same directory as the program you can run the command from the terminal with any combination of options. I have listed some examples below with their corresponding charts.

1. python homework\_0.py -l -s -sw -ph "the force" the\_original\_trilogy.txt --find\_characters

A screenshot of a graph

Description automatically generated

***The results look squished, but can be resized to your computer screen. I just wanted to provide a complete picture of what this software does. Below is a better look the data above.***

***A graph with a line

Description automatically generated***

***A graph of a number of words

Description automatically generated***

1. python homework\_0.py the\_original\_trilogy.txt --lower --stem --stop\_word --find\_characters

A graph with green bars

Description automatically generated

A graph with a line going up

Description automatically generated

A graph of a number of words

Description automatically generated

**Analysis:**

**Fig A: Most common words without Stopword removal (lowered and stemmed)**

**Total number of tokens: 77155**

A graph of a graph and a graph of a graph

Description automatically generated

**Fig B: Most common words without Stopword and Character removal**

**Total number of tokens: 33781**

A graph of a graph of a graph

Description automatically generated with medium confidence

To make this analysis a little easier I've broken up the two searches I did above into smaller chunks to more easily demonstrate the effectiveness of normalization and how it significantly changes the datasets and final conclusions. The top figure shows a general analysis of the entire corpus without stop words removed. There's some notable things here are that the most frequently used words are all common words that don't really have any relevance to the particular data set. Not to mention, there are a lot of them. The word duh was mentioned almost 6000 times throughout all three of the original Star Wars scripts.

By removing the stop words and the common names we can exclude a massive amount of data. Figure a had over 77,000 tokens when including the stop words. By excluding the stopwords we were able to half the amount of tokens There were in the final analysis. This also yields data that is more specific to the data I analyzed. the most common word in this data set was the word “star” followed by “cockpit” and other words like “ship” and “rebel”. basically we can conclude that if somebody was looking at the raw data without having any of the stop words removed or the other characters that appear within a movie script, they simply wouldn't have been able to tell what the data they were analyzing pertained to.

I'm not sure why, but I also noticed that removing the stop words had an effect on the least common words as well. I'm not sure if this is because i took a small sample size when doing this demonstration, or if some words were removed/stemmed making them appear less or more. What is clear is that the words that do appear least often within the script appear to have 2 commonalities. They are either long, or have very little to do with the specifics of the Star Wars universe. Something that is important to note is that in figure B most of the uncommon words have clearly been stemmed and are meant to be longer words that have been cut off. I have a hunch that this is a symptom of using the built in stem function in the NLTK lib, so an easy way to remedy this would be to build a custom stemming function for this particular task.

Moving beyond the general analysis of the charts I have included in this report I see that there are some similarities column but also differences between them and the word frequency plots seen on the Zips law wiki page. The examples shown have a clear linear deviation for the frequency of words we're at the low frequency end the plot starts to take a staircase shape. My graphs do end up taking that staircase shape, but the entire shape as a whole is more of a gentle slope. I believe this has something to do with the what's the difference in sample size as well as vocabulary and some of the processing that went on in my graphs

A graph of a number of words

Description automatically generated

For me, this exercise gave insight into how Google or ChatGPT must work. They have to remove all the fluff words so that they can get a better context as to what a user is querying. It's quite cool when you consider that! The verbs and nouns that are contained within a corpus provide more context as to what the actual information is about. while we humans prefer to have modifiers in our sentences to make what we say more clear and make sense to us, search engines and other technologies use natural language processing to get rid of this information and get to the meat of the corpus

**Works Cited:**

I got really excited about this assignment and used a bunch of tools and references to complete it Including but not limited to:

* Our textbook
* The NLTK docs
* The Arparse docs
* The Pathlib docs
* GitHub Copilot
* W3 Python Resource
* My Dad (He’s a datascientist)
* ChatGPT for the several bugs + and the graph that uses log Scale