**Report for Homework One – Corpus Analysis – Connor Homayouni**

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

**Introduction to the data selected**:

For the previous HW I chose to use Movie scripts for normalization. My corpus this time consists of quotes from famous heroes and villains across pop-culture. I chose these two categories for several reasons: The quotes are very diverse. Some of the quotes from the villains sound more like hero quotes, and some of the things said by the heroes are done right dark. I was interested to see the final data and some of the probabilities for the words. It would be fairly easy to manipulate the quotes in python because of the formatting( I will explain that later). Lastly, I chose the dataset simply because I thought that it would be more fun to mess around with. Here is a sample of each of the categories below:

**Sample Hero Quotes:**

* **"With great power comes great responsibility." - Uncle Ben, Spider-Man (2002)**
* **"I'm just a girl, standing in front of a boy, asking him to love her." - Anna Scott, Notting Hill (1999)**
* **"I feel the need—the need for speed." - Maverick, Top Gun (1986)**
* **"To infinity and beyond!" - Buzz Lightyear, Toy Story (1995)**
* **"I could do this all day." - Steve Rogers/Captain America, Captain America: The First Avenger (2011)**

**Sample Villain Quotes:**

* **"It is unavoidable. It is your destiny. You - like your father - are now... mine!" –Emperor Sheev Palpatine/Darth Sidious (Star Wars: Return of the Jedi/Star Wars: The Rise of Skywalker)**
* **"Don't you turn your back on me, Harry Potter! I want you to look at me when I kill you! I want to see the light leave your eyes!" -Tom Marvolo Riddle/Lord Voldemort (Harry Potter)**
* **“The world is not what it ought to be. Humanity longs for the eternal for a world beyond time, because time is what enslaves us. Time is an insult. Death is an insult. Doctor, We don't seek to rule this world... We seek to save it.” -Kaecilius (Doctor Strange)**
* **"You've come to die! Your world is now my world, like all worlds!” -Dormammu (Doctor Strange)**
* **"You, on the other hand, will die with the Rebellion!” -Orson Krennic (Rogue One: A Star Wars Story)**

There are over a hundred quotes for each category. Please feel free to give them a look! The list used is an amalgamation of a few lists that I found online and my own memory. I compiled the two lists myself and imported them into 2 text files that are accessible through Github. One consideration I had to take into account was the difference amount of documents that are contained in each category. The villain quotes are shorter the hero quotes, but I was able to find more of them over time. I’ll talk about this in the sections below.

**How to use the script(Methodology):**

The development process for these scripts was rather organic and that is reflected in how I ended up developing two Python scripts for this homework assignment. originally I intended to fit all of these features into one script, and as a matter of fact the analysis should only require one script, but what ended up happening was that I developed the portion meant to calculate probability using naive bayes equations we learned in class about a week and a half prior to starting on the LDA portion of the homework assignment. The original script was getting too cluttered, so i decided to split the script separating it by its two major functions. Each script contains the same normalization options and data set, so the calculations made should be accurate for both.

To use my program to analyze the hero and villain quotes you simply have to download the program **NB.py, LDA.py** and the text files **“hero\_quotes”** + **“villain\_quotes”** and make sure that they are added to the same directory. I've built the scripts 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. You will have to download a significant amount of libraries to get the code to work. This was a big experimentation assignment for me and I tried a whole bunch of different functionalities from a lot of different libraries throughout this assignment. I believe for the next assignment that I will mostly be using the **gensim** and **numpy** libraires so that will be more straight forward in the future.

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

* **$** **python NB.py – custom**
* **$ python NB.py—normal**

I should also mention that there is 1/3 option when triggering my script. You can add **--data\_type** to change what method is used to vectorize the data (this feature only works on NB.py, the other has COUNT and TF-IDF hardcoded in) and then adding these phrases: ['count', 'binary', 'tfidf']

Depending on which option you choose it changes the normalization that is applied to the data set which in turn changes the outcome of their likelihood and ratios. For the best results I recommend using my custom option. It does things like removing names, special characters, lower casing everything, and then only grabbing the residual data that's leftover within the quotation marks to make up each document. You can see some of the example of the qoutes as they have been normalized and sorted into their documents below.

|  |  |
| --- | --- |
| **Sample hero quotes:** | **Sample villain quotes:** |
| great power great responsibility | unavoidable destiny father |
| girl standing boy love | turn harry potter kill light leave eye |
| feel speed | growl |
| infinity | butt head |
| day | fear death |
|  |  |

For the rest of the report I will mainly focus on the results that were computed using my custom normalization function seeing that the data is overall more accurate. To illustrate the difference I have plotted out the overall stats of the corpus after it has undergone normalization in both methods and been sorted into the BoW format.

**Standard NLTK Normalization**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Category** | **Num of Quotes** | **Num of Tokens** | **Num of Types** | **Average # of Tokens per Doc** |
| Hero | 339 | 1149 | 114 | 3.389380531 |
| Villain | 851 | 1277 | 285 | 1.500587544 |
| Total | 1190 | 2426 | 1775 | 2.038655462 |

Standard normalization will parse out the sentences, but not as full quotes. It also leaves in numbers, which drastically changes the number of tokens and types.

**Custom Normalization**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Category** | **Num of Quotes** | **Num of Tokens** | **Num of Types** | **Average # of Tokens per Doc** |
| Hero | 110 | 560 | 106 | 5.090909091 |
| Villain | 210 | 616 | 189 | 2.933333333 |
| Total | 320 | 1176 | 927 | 3.675 |

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.

**Results Analysis**

Our first task was to use the naive bayes model of probability to compute the probabilities of or its belonging to their categories. This was oddly simple and rather underwhelming considering that I only had two datasets that were fairly small. I'm sure we'll build on this in the future but what I wanted to do but ended up running out of time, was have it so that the user could type a sentence and the program good spit out probabilities on how likely it was to come from a hero or a villain. That's a project for another time though. I didn't want to bombard the user with too much information So what I did was I had A set of loops run five times each that gathered the likelihood and probability of the top five bottom 5 and a random 5 words from within the data set and calculate the likelihood that they were in the hero or villain category. A positive score for a category it was indicative of the word most likely coming from said category while a negative score means that it's less likely to come from that category. All of these calculations were made using laplace smoothing and I used log form to calculate the naive bayes probability. I couldn't really think of a good way to format these, so i just had them output to the terminal. You're able to recreate these simply by running the program on normal mode and custom mode, so feel free to try it out!

**Theses are snippets of the probabilities of the words after using –custom normalization**

A screenshot of a computer screen

Description automatically generated

**And as you can see Below, if you use just the standard NLTK normalization on this data set you really can't distinguish any useful information besides the probability of the random words. It gets over saturated by numbers, and small stop words that don't really matter.**

**A screenshot of a computer screen

Description automatically generated**

Now this data gets interesting once we perform the Latent Dirichlet Allocation. Just a quick disclaimer for this section I wanted to use the PyLDAvis library to create visuals for the topic modeling, however it seemed that the documents that were provided for this library where deprecated and I couldn't get it to work on my computer. What i ended up doing for this analysis was exporting the outputs of the LDA to a CSV and reformatting it in excel. You can see those diagrams below:

Distribution of the Topic likelihood based on Hero or Villain

|  |  |  |
| --- | --- | --- |
| **Model** | **Topic ID** | **Words** |
| COUNT | 0 | 0.041\*"man" + 0.032\*"life" + 0.023\*"kill" + 0.019\*"death" + 0.019\*"good" + 0.018\*"power" + 0.016\*"people" + 0.013\*"wanted" + 0.013\*"die" + 0.013\*"time" |
| COUNT | 1 | 0.027\*"time" + 0.021\*"day" + 0.018\*"roar" + 0.016\*"live" + 0.015\*"god" + 0.015\*"power" + 0.014\*"life" + 0.014\*"truth" + 0.012\*"heart" + 0.012\*"child" |
| TF-IDF | 0 | 0.019\*"kill" + 0.016\*"power" + 0.015\*"good" + 0.013\*"man" + 0.013\*"roar" + 0.012\*"god" + 0.012\*"life" + 0.012\*"day" + 0.010\*"gon" + 0.010\*"people" |
| TF-IDF | 1 | 0.022\*"roar" + 0.021\*"man" + 0.017\*"life" + 0.014\*"time" + 0.012\*"die" + 0.011\*"king" + 0.011\*"death" + 0.011\*"hero" + 0.011\*"day" + 0.010\*"truth" |

|  |  |  |
| --- | --- | --- |
| **Model** | **Score** | **Topic** |
| BOW | 0.7263821 | 0.041\*"man" + 0.032\*"life" + 0.023\*"kill" + 0.019\*"death" + 0.019\*"good" |
| BOW | 0.27361792 | 0.027\*"time" + 0.021\*"day" + 0.018\*"roar" + 0.016\*"live" + 0.015\*"god" |
| TF-IDF | 0.70195293 | 0.022\*"roar" + 0.021\*"man" + 0.017\*"life" + 0.014\*"time" + 0.012\*"die" |
| TF-IDF | 0.29804707 | 0.019\*"kill" + 0.016\*"power" + 0.015\*"good" + 0.013\*"man" + 0.013\*"roar" |

Now I know that the format is a little bit difficult to understand, but essentially each row shows the likelihood of that word belonging to that particular topic. these aren't all of the words that are available, but just a sample so that you can see their size. If you want to see how the probability for all the words in the corpus, you can modify the code to run the four loop until it reaches the end of the length of the corpus. There are only two topics available and they both correspond to hero quotes (topic 0) and villain quotes( topic 1). These are the results that we'll display in the terminal if you run my LDA.py program. What's interesting is that we can clearly see the differences in the methods simply using count from using TF-IDF.

Using counts simply factors in how much a word is used and its frequency within the corpus. If we take the count model of topic 0 for example we see that it's more likely that the word man be used and a heroes quote. But this doesn't really have an accurate barometer of what's actually covered and the heroes quotes. If we look closely we see that there are other words like life, kill, death, and good that also appear very frequently in that topic period now, if we take a look at the tfidf results for topic 0 we see that things change a little bit In words like kill, power, and good are more likely to appear and topic 0 while the likelihood of man appearing has declined significantly. This is because the TF-IDF approach ways the word frequencies by the measure of how much they appear and gives them a value. This makes it so that more common words across the documents have lower weights in the overall end results which can lead to more accurate/specific data at the end of our calculation.

Now I didn't think the diagrams I made word that entertaining, so I made these cloud graphs that better illustrate the concept I was writing about above in a more high level way. These graphs below correspond directly to the values of each of the words given the LDA analysis.

The bigger the word the more likely it is to appear in that category. On the left is topic 0, or hero quotes, and on the left is topic one, also referred to as villain quotes. This first analysis it's simply of the standard count bag of words method. Personally, I find it interesting that death and power were more frequently said in hero quotes rather than villain quotes. And taking a look at the villain quotes we see that there are a whole lot of words that can be taken as being positive in this context.

A close-up of words

Description automatically generated

Now delineating from that previous analysis we can now take a look at the TF-IDF. Already the data displayed seems to be more aligned with what I would expect it to be at least on the heroes side. The villain category still has a lot of words in it that I would consider to have a mostly positive connotation, but it's still interesting and the words have shifted significantly.

A close-up of words

Description automatically generated

Now, I know I've covered the data pretty thoroughly, but I've also included a bar chart below as another way of illustrating the results of the LDA. When you run my code you'll also see that there is an extra graph that displays the frequency of words and graphs the overall curve, but I didn't include that in this report because the graph itself was too big and it was too hard to read from within the document.

A screenshot of a graph

Description automatically generated

**Final discussion:**

**What I learned from the dataset** is that oftentimes the things that are iconic for heroes to say in pop culture are things that are inspiring oftentimes to the point of pushing a rebellion forwards or provoking action. This has led the words that are associated with the hero quotes to be more violent and kind of having an air of power to them. Whereas if you look at the data coming out about the villain quote you see that a lot of the words don't have any things to do with action against others they seem to be words that would be associated with general conversation, but that kind of makes sense considering that most of these quotes come from villains who have an established measure of power. I'm interested to see what other analysis could be applied to see exactly what sentences are most likely to be from a villain given the quotes that I have trained this model on.

**What I learned from the process** is that it's better to process a full set of data only after you have a fully fleshed out plan. I spent a lot of time spinning my wheels on this assignment because I would complete a section and then have to go back and redo the section because of a requirement that I didn't foresee. It should be elementary but I needed to read the full assignment sheet prior to beginning. Part of the difficulties wasn't all my fault though. LDA is a significant challenge and when you're working and manipulating so much data you're bound to run into trouble. I think some of the issues I ran into will be resolved by the next assignment. One of the things i would do differently next time would be to grab an open source data set from the web. I like what i chose because it was fun, but making sure everything was formatted proved to be rather difficult, and there were some problems that i had because my data set was relatively small.

**Works cited and acknowledgements:**

**There were quite a few references that I used to complete this assignment these include:**

* our textbook
* the professors slides
* w3schools.com
* [Topic Modeling Tutorial (Latent Dirichlet Allocation) in Python](https://www.youtube.com/watch?v=Y79sCtzddyA)
* Python for Dummies
* Github Copilot