Peyton Skwarczynski

CSI 4180 – Natural Language Processing

Dr. Steven Wilson

NLP Homework 1: Corpus Analysis

1. Dataset:

The dataset I chose was the "yelp_review_full" dataset on huggingface, that consists of various yelp reviews and their associated star ratings on a scale of one to five stars. I thought this dataset would be intriguing to analyze since it provides documents that are written by the everyday individual, rather than a highly edited and structured document like a news source or movie script. This data splits each yelp review into an individual document, with the categories being the associated star review that the person gave.

Category:	Average Number of Tokens:	Number of Documents:
1 Star Reviews	73.40	189
2 Star Reviews	78.03	240
3 Star Reviews	79.12	208
4 Star Reviews	65.48	174
5 Star Reviews	62.70	189

The above average number of tokens per category is an output of my code. It shows that five-star reviews tend to be much shorter than lower-star reviews. This makes a lot of sense, as it seems plausible that people who are writing five-star reviews do not have much to say other than that the company is excellent, whereas people writing poor reviews can go on forever about their horrible experience at the particular establishment. The output also shows that there is a similar amount of documents in each category, allowing the data to be equally distributed amongst every level of review.

Even though the dataset contained 650 thousand yelp reviews, I kept my main analyzation to 1 thousand reviews due to inconclusive outcome data from the Naïve Bayes log likelihood ratio when working with extremely large data sets. When trying to produce a list of the top 10 words sorted by their log likelihood ratios, and using large data sets, the majority of the words printed out would be names of people, which does not show anything conclusive about how words relate to a specific star rating. The printed names could either be representative of employees or businesses that are named after people, but nonetheless, names do not give any further insight on this topic, so I decided to stick to a smaller data set. When limiting the dataset to a smaller amount of yelp reviews, I was able to get rid of the names in the log likelihood ratio.

2. Methodology:

For my initial text normalization strategy, I focused on reimplementing the normalization styles I had utilized in the previous Counting Tokens assignment. Since those formats of text preprocessing worked well in the previous assignment, I had initially thought that all I would need to do is implement the same steps I already completed. However, as I started to get further along into the coding process,

where I was printing log likelihood ratios and LDA outputs, I started to output strings that were not actually words, which forced me to take a look back at my normalization strategy.

Initially, I ran the dataset through lowercasing, punctuation removal, and token lemmatization. These steps by themselves proved to be quite effective. Yelp, like other social media applications, does not have any editor software that analyzes and corrects grammatical issues with posts before they are submitted. This allows people to use any forms of punctuation and capitalization techniques, even if they are not grammatically correct. Even with this initial text preprocessing, my program still output strings that were not actually words. I realized that these misspelled words contained a random amount of 'n' characters concatenated onto the front of the words. Looking further into the dataset, I figured out that the original documents also contained newline characters "\n". During tokenization, these newline characters would be added to the string following the newline, and since all punctuation within the documents were removed, this would add additional 'n' characters to the front of random strings. To account for this, I implemented an additional text preprocessing step that removes all newline characters before punctuation removal. This solution was effectively able to remove all the misspelled words in my output that contained a random amount of 'n' characters at the front of the words.

For my additional experimentation form of text preprocessing, I chose to implement stop word removal. Before adding the stop word removal step to preprocessing, the LDA output would be filled with stop words that contained zero context to the topic they were supposed to be representing. This made it impossible to decipher topics from one another. After adding in stop word removal, the LDA output was filled with words that contained more context to each topic, allowing reasonable conclusions to be made regarding the subject of each topic calculated. With smaller data sets, I noticed that the LDA output for each topic is more general, and thus harder to make subject conclusions, compared to looking at larger data sets.

3. Results & Analysis:

(All screenshots taken after copying and pasting output from terminal to text file)

```
Top 10 words in the label: 1 Star Reviews
('extraction', 5.823913786817824)
('dental', 5.7286036070135)
('virus', 5.7286036070135)
('groomsman', 5.50546005569929)
('approximately', 5.371928663074767)
('emmert', 5.371928663074767)
('wearhouse', 5.371928663074767)
('amanda', 5.371928663074767)
('lock', 5.217777983247508)
('mildew', 5.217777983247508)

Top 10 words in the label: 2 Star Reviews
('inconsistent', 5.6243700217524975)
('deck', 5.490838629127975)
('popcorn', 5.490838629127975)
('brugge', 5.336687949300716)
('junction', 5.336687949300716)
('loaded', 5.336687949300716)
('ipa', 5.154366392506763)
('throw', 5.154366392506763)
('trout', 5.154366392506763)
('dormont', 5.154366392506763)

Top 10 words in the label: 3 Star Reviews
('tom', 5.844000509361145)
('bistroid', 5.487325565422411)
('vacuum', 5.333174885595154)
('caffe', 5.333174885595154)
('cider', 5.150853328801199)
('knock', 5.150853328801199)
('cheez', 5.150853328801199)
('halfprice', 4.927709777486989)
('spectre', 4.927709777486989)
('minutellos', 4.927709777486989)
```

```
Top 10 words in the label: 4 Star Reviews
('yum', 5.1639904049189 38)
('hat', 4.940846853604729)
('tortelloni', 4.940846853604729)
('yay', 4.940846853604729)
('walleye', 4.940846853604729)
('deliciously', 4.940846853604729)
('cardio', 4.940846853604729)
('twist', 4.940846853604729)
('pirate', 4.940846853604729)
('benedict', 4.940846853604729)

Top 10 words in the label: 5 Star Reviews
('walter', 6.162222977236242)
('mattress', 5.852068048932403)
('pierogi', 5.533614317813868)
('shadow', 5.400082925189345)
('chiropractic', 5.063610688568133)
('smiley', 5.063610688568133)
('adopting', 5.063610688568133)
('bicycle', 5.063610688568133)
('headlight', 4.840467137253922)
('registry', 4.840467137253922)
```

The above screenshots are the result of the Naïve Bayes log likelihood ratio to determine the top ten words that most associate with the different categories. These results were output when using 1 thousand total yelp reviews, or in this case, documents, in order to minimize the number of names displayed as most correlated to a specific category. As mentioned above, when using larger data sets, up to 15 thousand yelp reviews, these outputs would be filled with people's or businesses' names, rather than any verbs or adjectives that can be better analyzed. Even when looking at a smaller number of documents, surprisingly, the output of this log likelihood ratio still shows very minimal adjectives. I initially thought that the most common words to be found in reviews would be words that one would use to describe whatever they would be rating. For example, if looking at one-star reviews, I expected to find words like "horrendous" or "terrible". On the other hand, if analyzing five-star reviews, I predicted to see words like "amazing" or "excellent" to describe the facility or service. My expected outcome only occurred within the four-star reviews, where some of the most related words included "yum", "yay", and "deliciously". These adjectives quite clearly relate well to four-star reviews, as yelp reviewers would be complementing the services in high graded reviews. In reality, however, a majority of these output words represent the overall product or service that is being provided, instead of the quality of the service. In one-star reviews, some of the most common words include "extraction", "virus", and "dental". These tokens all point towards healthcare service, which shows that this specific dataset, or perhaps the specific document size I chose, comprises mainly of poor reviews for healthcare facilities. The same is true for five-star reviews, as some of the most common words include "mattress", "pierogi", and "chiropractic", which all point to a specific service, rather than the exact quality of an arbitrary product or service. I would have liked to analyze the results of this in a larger dataset, but without being able to efficiently preprocess out names of people and businesses, as previously mentioned, this would be hard to accomplish.

```
Topic: 0
Words: 0.009*"good" + 0.007*"get" + 0.006*"food" + 0.005*"place" + 0.005*"really" + 0.005*"one" + 0.004*"also" + 0.004*"like" + 0.004*"bar" + 0.004*"would"

Topic: 1
Words: 0.010*"place" + 0.008*"food" + 0.008*"good" + 0.007*"time" + 0.006*"one" + 0.005*"would" + 0.005*"go" + 0.004*"lot" + 0.004*"get" + 0.004*"always"

Topic: 2
Words: 0.008*"good" + 0.007*"food" + 0.006*"place" + 0.006*"one" + 0.005*"really" + 0.005*"even" + 0.005*"time" + 0.005*"store" + 0.005*"service" + 0.004*"ordered"

Topic: 3
Words: 0.008*"get" + 0.006*"time" + 0.006*"like" + 0.006*"one" + 0.006*"place" + 0.005*"good" + 0.005*"dont" + 0.004*"go" + 0.004*"food" + 0.004*"service"

Topic: 4
Words: 0.008*"good" + 0.008*"food" + 0.006*"like" + 0.005*"also" + 0.005*"pizza" + 0.004*"great" + 0.004*"time" + 0.004*"get" + 0.004*"go" + 0.004*"one"

Topic: 5
Words: 0.009*"time" + 0.008*"get" + 0.007*"like" + 0.007*"food" + 0.007*"good" + 0.007*"place" + 0.006*"would" + 0.005*"go" + 0.005*"one" + 0.005*"dont"

Topic: 6
Words: 0.018*"food" + 0.011*"place" + 0.009*"one" + 0.009*"good" + 0.008*"like" + 0.007*"get" + 0.006*"time" + 0.006*"go" + 0.005*"back" + 0.005*"even"

Topic: 7
Words: 0.010*"good" + 0.009*"place" + 0.007*"get" + 0.006*"back" + 0.006*"food" + 0.006*"like" + 0.006*"pizza" + 0.006*"one" + 0.005*"time" + 0.005*"would"

Topic: 8
Words: 0.009*"place" + 0.007*"food" + 0.007*"would" + 0.006*"like" + 0.005*"great" + 0.005*"time" + 0.005*"get" + 0.005*"one" + 0.005*"restaurant" + 0.005*"go"

Topic: 9
Words: 0.014*"food" + 0.009*"time" + 0.008*"good" + 0.007*"great" + 0.006*"get" + 0.005*"service" + 0.005*"like" + 0.004*"bar" + 0.004*"one" + 0.004*"place"

The top three topics for each category:
Label 0 (1 Star Reviews): 6, 3, 7
Label 1 (2 Star Reviews): 6, 5, 7
Label 2 (3 Star Reviews): 7, 6, 5
Label 3 (4 Star Reviews): 6, 7, 5
Label 4 (5 Star Reviews): 6, 5, 0
```

The above screenshot is the output from LDA topic modeling. The first part of the image is the result of calculating 10 different topics and the top 10 words, along with their probabilities, that relate back to the arbitrary topic. The second part of the output selects the most related topics to each of the 5 original categories. When analyzing results from a small number of documents, as seen above, there are enough quality words to gather a subject for each category, but there seems to be little difference between the various topics. For example, even though topics 5 and 6 can clearly be about a particular place to get food, the second part of the LDA analysis states that these topics can be found in almost every single category. Since one of the words in each of the two topics is "good", how can the word "good" be representative of both a five-star and one-star review?

Overall, each of these topics should have a different subject, but this cannot be determined because the results say that a single topic, like topics 5 and 6, can be representative of almost every single category. Unless every yelp review in the dataset is about a single product, then this should not be the case. However, this idea is actually ruled out entirely due to the previous Naïve Bayes analysis. During the log likelihood calculations, we were able to conclude that there are a variety of different services that are included within the dataset, as seen through the various related words to each category. The main reason why I believe this undetermined result is being computed is due to the preprocessing step of stop word removal. Even though this step is essential to remove words like "I", "and", and "a" that provide no meaning, it will also remove the word "not". From personal experience, when writing a negative review about a company or service, I will refrain from using descriptive adjectives as a way to save my time typing out the review. From my point of view, I would want to express my unhappiness in as short of time as possible, so I would tend to use the word "not" to negate any simple, positive adjective, such as "good". Using this ideology, some of the topics should actually be describing companies that receive high negative reviews, but it is impossible to determine this due to the removal of the word "not" in stop word preprocessing. When removing this stop word from the dataset entirely, the topics can seem to be extremely similar to one another, even though this is not actually the case. If we were able to include the word "not", I believe that some of the topics would then include that stop word, which would then relate that topic more towards only the negative reviews. Even though some of the data is inconclusive, there still exists some topics that can be properly identified. For example, one topic that can be correctly described is topic 0. This topic uses words like "good", "food", "place", and "bar", and most closely relates to the five-star review category. Topic 0 is then an obvious representation of a subject that is similar to a highly reviewed restaurant/bar.

Additionally, it is important to note that when testing the LDA analysis with higher amounts of yelp reviews, the results will display slightly more words that can be used to determine the topic subject. For example, instead of only being able to determine that the subject of a topic is restaurant, other words in the topic would suggest that it is more specifically a pizza or Chinese restaurant.

4. Discussion:

The main thing I learned about my dataset is that people rarely tend to use harsh, negative adjectives when writing poor yelp reviews, but will use positive adjectives when writing highly rated reviews. Rather than using harsh terminology to describe whatever they are reviewing, it can be seen that people tend to focus on the poor experience and service provided. Even though some negative reviews can be long winded, these focus much more on the customer's individual experiences, which not

only are hard to replicate in another customer's review, but also focus more on warning away other customers from the business due to their poor experience, rather than bashing the product entirely. On the other hand, when writing positive reviews, people are more likely to use positive adjectives to describe the business but are also more likely to write much shorter reviews. People that give out five-star reviews can feel like the review itself is enough feedback, so they will not write much in their post, if anything at all.

Since I am still extremely new to python programming, the most challenging part of this assignment, and also what I learned most from this project, was exactly that. I did not think it was hard to understand the Naïve Bayes and LDA models when we were going through them during class, but it was a completely different story when trying to code this assignment. I found it quite difficult for me to conceptualize how to format the bag of words model in python code, as well as the Naïve Bayes log likelihood ratios. Additionally, some of the python data structures are still confusing to me, especially the included functions to access them. In order to figure out the best method of implementation, I had to cross reference many different online python sources. Another lesson that I learned throughout the completion of this assignment was being more open to analyze whatever results were returned from my code. Starting with the topic of yelp reviews, I had an idea from the beginning what my output should be roughly similar to. When I ended up receiving values that are far from what I thought it should be, I would first think that my code is incorrect and look to make changes, rather than analyzing why these results could be occurring. It took me a while to step back and actually make this analysis, rather than immediately thinking that something is wrong with my code.