Liam Abbott

SI 601, Winter 2015, Project Report

Motivation:

For my project, I was interested in analyzing the text of online user product reviews to count the most frequently used two-word phrases in both positive and negative reviews. I wished to compare the frequency with which these words were used in reviews with the frequency of their use in the English language, as measured by Google Books' n-gram corpora. I originally planned to look at user reviews of Amazon products, but the data source I intended to use was made unavailable between the time of my proposal and the time of this report. So instead of analyzing Amazon product reviews, I analyzed Yelp user reviews of businesses. To add a layer of complexity to the project, I focused my analysis specifically on businesses categorized as bars. My idea was that if certain phrases are used more frequently by users who leave positive reviews for bars, then bar owners should focus on marketing the aspects of their bars associated with those phrases. Alternatively, bar owners should seek to avoid problems associated with phrases frequently used by users leaving negative reviews.

Data Sources:

The first two datasets I used are the 'business' and 'review' JSON files released by Yelp as part of their Dataset Challenge, which can be download from http://www.yelp.com/dataset_challenge after agreeing to the Yelp Dataset Challenge Terms of Use. The 'business' JSON file is 54 MB in size. It contains various attributes for each business, but the attributes my analysis used were 'business_id', 'stars', and 'categories'. The 'review' dataset is 1.4 GB in size and contains various attributes relating to 1.6 million reviews of those business by Yelp users. The variables from the 'review' dataset I used were the business id of the business being reviewed, the stars given to the business by the reviewer, and the text of the review. My analysis focused on the 3,628 businesses classified as bars in the 'business' dataset. These bars had a total of 182,416 reviews by Yelp users. The 'business' and 'review' JSON files submitted

with this project are just samples of the first 5000 records found in the full files, which can be accessed via the link above.

The second dataset I used in my analysis is the Google Books n-gram corpora. Specifically, my analysis used the bigrams from the American English Version 2 corpus. Because the dataset is so large (just the American English bigrams dataset is 38 GB compressed) and my use for it in this analysis was a simple comparison of frequency of use for certain selected bigrams, it didn't make sense for me to work with the entire dataset. Instead, I used the frequency of occurrence data for certain phrases drawn from the Google Ngram Viewer (https://books.google.com/ngrams).

To incorporate the data from the Viewer into my analysis, I imported a cool Python script (getngrams.py) written by Matt Nicklay (GitHub username econpy) into my own analysis script. I downloaded his getngrams.py script from the public GitHub repository: https://www.github.com/econpy/google-ngrams. This script is based on code originally written by the Culturomics team at Harvard University. A description of the original Culturomics code can be found at http://www.culturomics.org/Resources/get-ngrams.

Data Manipulation Methods:

From the Yelp datasets, the data I needed to begin my word count analysis were the star rating and review text of user reviews only of businesses categorized as bars. Because the 'review' JSON file contains only the business id of businesses being reviewed and not the business categories, I needed to work with both the 'business' and 'review' JSON files. To combine the data in these two files and extract the data I needed, I used the sqlite3 module in Python to create a sqlite database on disk.

My first step was to create a generator function that opened the 'business' dataset and yielded the business id and star rating for each of the businesses categorized as bars. I then created a second generator function that opened the 'review' dataset and yielded the id of the business being reviewed, the stars given, and the review text for **all** reviews in the dataset. As mentioned above, because business category was not an attribute in the 'academic' dataset, I was unable to filter out the non-bar reviews in this step. I used

generator functions to open and parse each of the Yelp datasets in order to avoid locking up a large chunk of memory by putting the data in Python lists, particularly the 1.4 GB 'review' dataset.

I then used the two generator functions to create two tables, 'business' and 'review', in a sqlite database on disk. After the tables were created, I joined the two on shared business id to create a table with 4 attributes (business id, total star rating, review star rating, review text) and 182,416 entries. Each of the entries in this joined table corresponded to a user review of a business categorized as a bar. From this table, I extracted the review star rating and review text for each entry. The final step in getting the data ready for analysis was separating the positive and negative reviews. I created a positive review list containing the review text of all 5-star reviews and a negative review list containing the text of all reviews with 2 or fewer stars. This is the information from the Yelp datasets I needed to begin my word count analysis.

Note on project code: I split the code for this project into two scripts. The first script (si601_project_preprocess_leabbott.py) takes the two Yelp JSON datasets as input, processes them as outlined above, and outputs 'pos_reviews.txt' and 'neg_reviews.txt' files containing the star rating and review text for all positive and negative reviews of bars. This preprocessing step requires almost 3 GB of hard drive space (1.4 GB for 'academic' JSON file and another 1.4 GB for sqlite database), so I wanted to leave the reader the option of starting from the analysis stage with the positive and negative review files. The second script (si601_project_analysis_leabbott.py) takes these text files as input and conducts the word count analysis that is the focus of this project. There is a third Python script used in my project (getngrams.py) that is not written by me (discussed above) and is imported and used by my analysis script.

Analysis and Visualization:

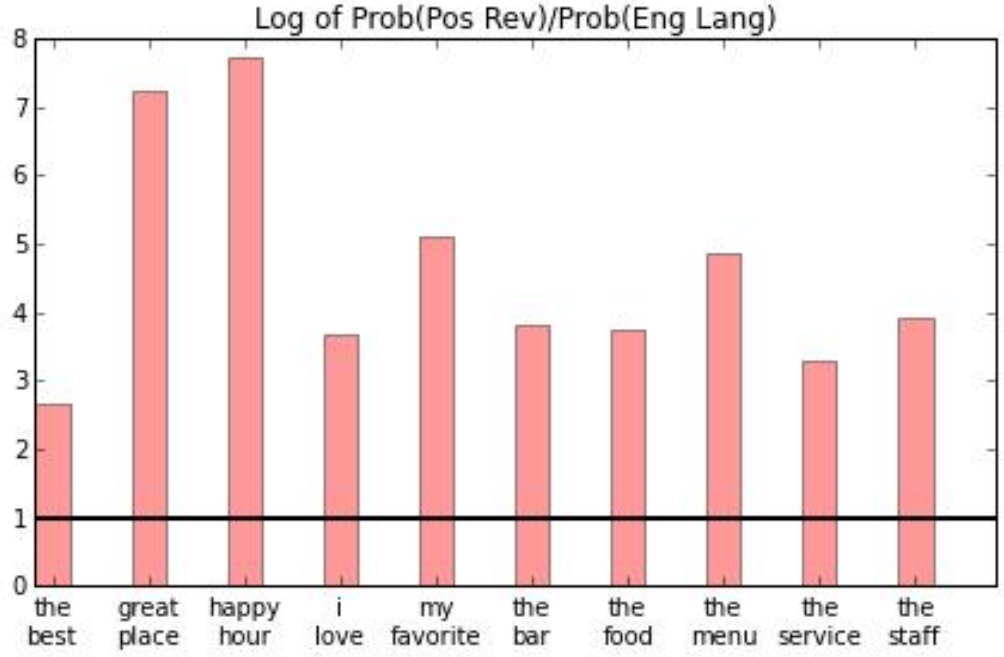
The analysis script of my project begins with opening the positive reviews text file described above. The script then creates a list of lists for positive reviews, where each sub-list consists of a bigram and the number of times it occurs in all positive reviews, with the master list sorted by number of occurrences of each bigram. Using the number

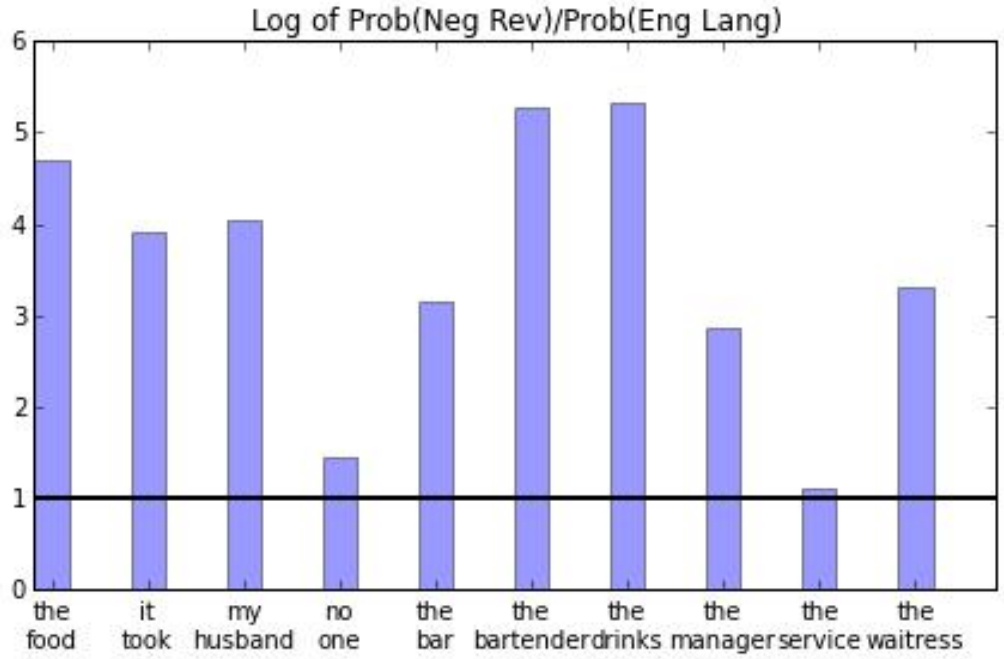
of occurrences, the script calculates the probability of each bigram as the number of times it occurred divided by the total number of bigram occurrences in all positive reviews. Then, another list of lists is created, where each sub-list now contains a bigram, the number of occurrences of the bigram, and the probability of occurrence for the bigram. This list of lists is written to a tab-delimited .txt file, 'pos_bgs_cts_probs.txt'. This entire process is repeated for negative reviews, resulting in a second .txt file 'neg_bgs_cts_probs.txt'. Both of these text files are sorted by number of occurrences in either positive or negative reviews, respectively, which is equivalent to the probability of occurrence.

The next part of my analysis involved comparing the probability of occurrence for bigrams in positive and negative reviews to the probability of occurrence for those same bigrams in the English language. To measure the probability of occurrence in the English language, I used the data provided by the Google Ngram Viewer (https://books.google.com/ngrams). Rather than searching for each bigram I was interested in using the Google Viewer, I used a Python script (not my script, see citation above) that extracts the frequency of occurrence data that is plotted in the Google Viewer for whichever bigrams are passed to it and saves the data in a .csv file. I used this script (getngrams.py) in my analysis by importing it into my own analysis script. My analysis script takes a string of bigrams picked by me and passes these bigrams (along with a set of parameters) to the function imported from getngrams.py. That function then creates a .csv file in the project folder containing the frequency of use data from the Google Ngram Viewer. My analysis script then reads in the data from the .csv file.

For the purposes of my project, I did not analyze all bigrams in the positive and negative bar reviews, but only a subset of the interesting phrases that were among the most frequently used in positive reviews or negative reviews. The bigrams I chose were descriptive or clearly associated with some aspect of the bar ('the food', 'the menu', etc.) rather than commonly occurring, but uninteresting bigrams ('of the', 'and the', etc.). I chose to analyze ten of the most commonly occurring bigrams in positive reviews and ten of the most commonly occurring bigrams in negative reviews.

After extracting the English language probability information for the selected bigrams, I computed the log of the ratio of the probability of a bigram occurring in a review to the probability of that bigram occurring in the English language. I took the log to normalize the scale of the ratios so the probabilities could be plotted on the same chart. After calculating these log-ratios, I used the matplotlib package to create and save the following plots (saved as 'pos_probs_chart.pdf' and 'neg_probs_chart.pdf'):





My analysis and summary plots show how certain bigrams are used more frequently in Yelp user reviews of bars than they are in the general English language. Certain topics ('the food', 'the bar') evoke strong feelings in many reviewers, as they appear frequently in both positive and negative reviews. Other topics ('happy hour', 'I love') are more common in positive reviews, while still more topics ('the manager', 'the waitress') are frequently seen in negative reviews.

Overall, I was pleased with the results of my project. It's certainly an imperfect and rough analysis, but I think the framework is useful in extracting certain insights into the vocabulary choice of bar reviewers. If I were to work further on this same project, I would like to analyze more bigrams than just the twenty that I hand-selected here. I would also like to analyze other n-grams, especially longer ones such as 3- and 4-grams that might contain more unique and interesting phrases than just the bigrams that I analyzed.