Text Analytics Homework 3
Rachel Rosenberg
November 2019

Part 1: Dataset Stats
Dataset description:
This dataset is a large corpus of Amazon reviews for kitchen & home products. It comes from the Computer Science department at UCSD.
It contains 499815 unique reviews, among a subset of only users who have more than 5 reviews on the site overall (to clean the data).
The original data is labeled with a number of stars, 1-5. When cleaning, I assign "good" reviews as those with 3+ stars and 'bad' reviews as those with one or two stars.

There are two classes in the cleaned dataset: good review (1) and bad review (0).
Class distributions below:
Label = 0: (451905, 2)
Label = 1: (47910, 2)
There is some class imbalance; this will be addressed in the models.

There are 93.605 average words in each document in the corpus.

The above is output from code in the dataset_stats.py script.

Part 2: Logistic Regression
For this experiment I varied the number of ngrams (1 or 2), the penalty or norm (L1 or L2) used i n regularization, and the Min_DF for TF-
IDF, which controls the number of features created. I used Accuracy and ROC score for my repo rting metrics - Accuracy because I want to know how the model performs as far as the misclassi fication rate, and ROC because it is a measure of how well the model performs on an unbalance d dataset, and this dataset was fairly unbalanced with more positive (3+ star) reviews than nega tive (1-2 star) reviews.

I took the following preprocessing steps:
* Labelling reviews with 3+ stars "positive" (1) and reviews with 1 or 2 stars "negative" (0).
* Removing all numbers from the text
* Removing all punctuation from the text
* Removing all stopwords (English) from the text
* Stemming all words

I used the following parameters identically for all models:

* Classes balanced. I did not want the model to predict based on the probability of a positive r eview, so I set class_weights to balanced in the Logistic Regression parameters.
* Latin encoding. All

* Max_DF of 0.3 (threshold for ignoring corpus-specific stop words) - I set this value fairly low because I did not want to ignore too many words. Since this is a very varied dataset, there will be many words that are important to context and also are repeated many times.
* Binary = False (items are counted in the number that they appear). Since there will be some important words that are used very few times, I didn't want them to be overshadowed (or considered equal to) words that are less important but used many times.

All models performed decently well, with accuracies ranging from 0.73 to 0.81 and ROC values ranging from 0.83 to 0.89. The best-performing model on both metrics used L2 regularization, a feature size of 500, and 2 ngrams. This is likely because the model was given more information to work with (both with bigrams and more features) and L2 regularization was used to cut the model to include only useful features.

Overall, the model could likely be improved upon if I used more features, more n-grams, and a lower max_df, but all of these things would increase training time too much to be scalable.

Part 3: Support Vector Machine

Part 4: FastText Classifier

Part 5: CNN Classifier

Part 6: "Predict" Script