DS-GA 1015, Text as Data
Prof. Arthur Spirling
Assignment date: March 12, 2019

Homework 2

This homework must be returned to Pedro Rodriguez' mailbox (2nd floor, 19 West 4th Street) by **5pm, April 2, 2019**. Late work will incur penalties of the equivalent of one third of a letter grade per day late.

It must be your own work, and your own work only—you must not copy anyone's work, or allow them to copy yours. This extends to writing code. You may consult with others, but when you write up, you must do so alone.

Your homework submission must be in one of the following formats: (1) A set of answers and a clearly commented `R` code appendix (**use comments to identify code relevant to each answer you produced**), (2) A report consisting of clearly marked answers, each accompanied by the relevant code (e.g., a report generated using `rmarkdown`, `knitr`, or similar). **In either case, your code must be included in full, such that your understanding of the problems can be assessed. Homework that does not conform to these guidelines will not be graded.**

You must turn in a paper copy: **no electronic copies will be accepted**.

Part 1

1. We would like you to perform some Naive Bayes classification **by hand** (that is, you may use math functions or DFM-creating functions, but not any built-in naive Bayes functions). Make sure to show your work!

   (a) Imagine a situation in which you receive emails from the two main U.S. parties in anticipation of the 2020 election. The contents of those emails after all relevant preprocessing are displayed in Table 1. Using the standard Naive Bayes classifier without smoothing, estimate for each party the posterior probability (or rather, the prior multiplied by the likelihood) that the following email was sent by the respective party: "immigration voter aliens help economy". Report these estimates. Based on these results, which party would you predict sent the mystery email? Explain whether you trust your findings and why.

   (b) Now impose Laplace smoothing on the problem and re-estimate each party's respective posterior probability. Report your findings. Based on these new results, which party would you predict sent the mystery email? Beyond computational reasons (i.e. avoiding $log(0)$'s), can you think of any theoretical reason why smoothing might make sense (hint: the above data is but a sample of each party's shared language).

email	content
republican1	immigration aliens wall emergency country
republican2	voter economy president growth security
republican3	healthcare cost socialism unfair help
democrat1	immigration country diversity help security
democrat2	healthcare universal preconditions unfair help
democrat3	economy inequality opportunity voter help
democrat4	abortion choice right women help

Table 1: Training set of presidential candidate emails.

Part 2

For this exercise you will use a database of Yelp reviews gathered for a Kaggle challenge (source). Each user left a star rating of 1-5 along with a written review. You'll be asked to use some of the supervised learning techniques we've discussed in class to analyze these texts.

Download the most recent version from the course GitHub. The data are available in the file "yelp_.csv".

Before we get started, be sure to actually read a few of the reviews, to get a feel for the language used, and any potential imperfections in the text created during the scraping process.

For each task (3) through (6), begin with the raw version of the text, and briefly explain which pre-processing steps are appropriate for that particular task.

2. Before we apply any classification algorithms to the Yelp reviews, we will need a general classifier that tells us whether the review was positive or negative—also referred to as the "actual score."

   (a) Divide the reviews at the empirical median score and assign each review a label as being "positive"—if the user score was greater than the empirical median score—or "negative"—if the review is less than or equal to the empirical median (you can use "1" and "0" as labels if you prefer, just be consistent as you do the exercises below).

   (b) For some tasks, we will need "anchor" texts at the extreme of the distribution. Create a character variable (name it "anchor") that has value "positive" if the user star rating given to a review is equal to 5, "neutral" if the user rating is less than 5 but greater than 1 and finally "negative" if the user rating is equal to 1. Report the proportion of reviews that are anchor positive, neutral and negative.

3. The first method we'll use to classify reviews as being positive or negative will be dictionary based. To do so, you will use the dictionaries of positive and negative words discussed in Hu

& Liu (2004)—available on GitHub. You must use the dictionaries provided and may not use any substitutes from R packages.

(a) First, generate a sentiment score for each review based on the number of positive words minus the number of negative words. Then, create a vector of dichotomous variables, of equal length to the number of reviews, in which texts that have a positive sentiment score are labeled "positive," while those with a negative score are labeled "negative"; if any of them have a sentiment score of 0, score them as positive. Report the percent of reviews in each category, and discuss the results.

(b) Create a histogram to visualize the distribution of the continuous sentiment measure. Your answer should be a graph.

(c) Evaluate the performance of your model at identifying positive or negative reviews by creating a confusion matrix with the positive and negative values assigned by the sentiment score (created in 3(a)) on the vertical axis and the binary "true" classifications (created in 2(a)) on the horizontal axis. Use this confusion matrix to compute the accuracy, precision, recall and F1 score of the sentiment classifier. Report these findings along with the confusion matrix. In terms of accuracy, how would you evaluate the performance of this classifier? (Hint: is there a baseline we can compare it to?)

(d) Use the non-anchor texts for the following task as we will be comparing the result with that obtained using wordscores. Use the predicted sentiment score to rank the reviews, where 1 is the most positive review and N is the most negative. Do the same using the actual sentiment score (the original star rating). Compute the sum of all of the absolute differences between the predicted rank and the actual rank of each review (see RankSum represented in Equation 1). Report your findings.

$$\text{RankSum} = \sum_{i=1}^{N} |\text{PredictedRank}_i - \text{TrueRank}_i| \tag{1}$$

4. Next, we'll train a Naive Bayes classifier to predict if a review is positive or negative.

(a) Use the "textmodel" function in quanteda to train a *smoothed* Naive Bayes classifier with uniform priors, using 80% of the reviews in the training set and 20% in the test set (Note: features in the test set should match the set of features in the training set. See quanteda's `dfm_match` function.). Report the accuracy, precision, recall and F1 score of your predictions. Include the confusion matrix in your answer.

(b) Were you to change the priors from "uniform" to "docfreq," would you expect this to change the performance of Naive Bayes predictions? Why? Re-estimate Naive Bayes with the "docfreq" prior and report the accuracy, precision, recall and F1 score of these new results. Include the confusion matrix in your answer. In terms of accuracy, how would you evaluate the performance of this classifier?

(c) How is accuracy affected if you fit the model without smoothing? Why might this be?

(d) In the above exercise we only used words as features. Can you think of other features beyond words that may help classify the sentiment of a document?

5. Although there isn't really an "ideology" in this example, there is an underlying positive-negative latent space that we can use to train a "wordscores model." In this question, you will be implementing "wordscores" by hand. You may use functions in base R and quanteda, but not the built-in "wordscores" function.

   (a) Create a vector of "wordscores" for the words that appear in the "anchor negative" and "anchor positive" reviews (from question 2b) using the technique described in Laver, Benoit & Garry (2003). That is, you should fit a "wordscores" model to the anchor texts. What are the most extreme words (i.e. words with the lowest and highest word-scores)? Report your findings.

   (b) Apply your wordscores model to the non-anchor documents. This should generate a wordscores estimate for each document. Calculate the RankSum statistic (described in Equation 1) of the reviews as scored by wordscores in the same way as you did for the dictionaries. Report your findings. By this metric, which did better: dictionaries or wordscores?

6. Now we'll attempt to do our best on the classification task using a Support Vector Machine (SVM). Since SVM functions are computationally intensive, restrict your analysis to the first 1000 reviews using the original ordering of the review data.

   (a) Describe an advantage offered by SVM or Naive Bayes relative to the dictionary approach or wordscores in classifying positive and negative reviews.

   (b) Using the "cross_validate" command, train an SVM. Your goal is to maximize out of sample accuracy with 5-fold cross-validation. Optimize over the relative size of the training and test sets by training an SVM in 10% intervals from 10% to 90% percent of the data. In other words, you should train 10 different models with 5-fold cross-validation. Report the mean accuracy for each model. Report which model results in the highest average accuracy for out-of-sample predictions made on the test set. In terms of accuracy, how would you evaluate the performance of this classifier?

(c) Take a guess as to which kernel would be best to use in this context, and discuss what assumptions about the data cause you to make that choice. Try both the radial and linear kernels; were you correct?

7. Finally, let's try out a Random forest classifier. For this question use the first 500 reviews in the dataset.

   (a) As we did for the Naive Bayes model, split the dataset into a training (80%) and a test set (20%) and construct a document feature matrix for each (Note: features in the test set should match the set of features in the training set).

   (b) Using the `randomForest` package fit a random forest model to the training set using the package's default values for `ntree` and `mtry` (also, set the `importance` argument to TRUE). Having fitted the model, extract the *mean decrease in Gini* importance for the feature set and order from most important to least important. What are the top 10 most important features according to this measure?

   (c) Using the fitted model, predict the sentiment values for the test set and report the confusion matrix along with accuracy, precision, recall and F1 score.

   (d) Now you will do some tuning of one the two model parameters. The package's default value for the argument `mtry` is $sqrt(\#offeatures)$. Estimate two more models, one for each of two different values of `mtry`: $0.5 * sqrt(\#offeatures)$ and $1.5 * sqrt(\#offeatures)$. As you did above, use each of the fitted models to predict the sentiment values for the test set and report the respective accuracy scores. Which value of `mtry` yielded the best accuracy?