DS-GA 1015, Text as Data
Prof. Arthur Spirling
Assignment date: March 10, 2020

Homework 2

This homework must be turned in on NYU Classes by **Tuesday, March 31, 2020, at 11pm**. Late work may be turned in up to 2 days late and 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 anyone 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 a PDF or HTML report, containing all written answers and code, generated from `RMarkdown`. **Raw `.R` or `.Rmd` files will not be accepted.**

Please remember the following:

- Each question part should be clearly labeled in your submission.
- Do not include written answers as code comments. We will not grade code comments.
- The code used to obtain the answer for each question part should accompany the written answer.
- **Your code must be included in full, such that your understanding of the problems can be assessed.**

Please consult "RMarkdown Basics" on the course GitHub for help with RMarkdown. You can also use the "Sample RMarkdown HW" template on the course GitHub to get started. Using this template is not required.

You should continue to use `quanteda` version 1.5.2 for this assignment.

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 (up to a proportionality constant: i.e.,

the prior multiplied by the likelihood) that the following email was sent by the respective parties: "immigration voter aliens help jobs". Report these estimates. Based on these results, which party would you predict sent the mystery email? Explain whether you trust your findings and why.

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

Table 1: Training set of presidential candidate emails.

(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 explain any theoretical reasons that smoothing would make sense (hint: the above data is but a sample of each party's shared language)?

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.

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—the true classificatiion.

(a) Divide the reviews at the empirical median star rating and assign each review a label as being "positive"—if the user rating was greater than or equal to the empirical median

score—or "negative"—if the rating is less than the empirical median (you can use "1" and "0" as labels if you prefer, just be consistent as you do the exercises below). These are your true class labels. Report the proportion that are positive and negative, and the median star rating.

(b) For some tasks, we will need "anchor" texts at the extreme of the distribution. Create a 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)—provided to you as "negative-words.txt" and "positive-words.txt". You must use the dictionaries provided and may not use any substitutes from R packages.

(a) Briefly explain which pre-processing steps are appropriate for this particular task.

(b) Generate a sentiment score for each review based on the number of positive words minus the number of negative words. Create a histogram to visualize the distribution of the continuous sentiment score.

(c) Create a vector of dichotomous variables, of equal length to the number of reviews, in which texts that have a positive sentiment score (from part (a)) are labeled "positive," while those with a negative score are labeled "negative"; if the sentiment score is equal to 0, score them as negative. Report the percent of reviews in each category, and discuss the results.

(d) 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(b)) 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?)

(e) Use the non-anchor texts for the following task (we will be comparing the result with that obtained using `wordscores`). Use the predicted sentiment score to rank the reviews, where rank 1 is the most positive review and N is the most negative.

Next, rank the non-anchor reviews by their star rating. You can handle ties any way you wish, but briefly state how you did so.

Compute the sum of all of the absolute differences between the predicted rank (from the sentiment score) and the star rating 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) Briefly explain which pre-processing steps are appropriate for this particular task.

   (b) Use the "textmodel" function in quanteda to train a *smoothed* Naive Bayes classifier with uniform priors, using 75% of the reviews in the training set and 25% 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.). To be clear, you should use +1 smoothing. Report the accuracy, precision, recall and F1 score of your predictions. Include the confusion matrix in your answer.

   (c) Were you to change the priors from "uniform" to "docfreq," would you expect this to change the performance of Naive Bayes predictions? Why? You are not required to fit a model for this question.

   (d) Re-estimate Naive Bayes with the "docfreq" prior and +1 smoothing. 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?

   (e) Fit the model without smoothing and a uniform prior. Report the accuracy, precision, recall and F1 score of your predictions. Include the confusion matrix in your answer. How does the accuracy compare to the previous models? Why might this be?

   (f) 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 10 most extreme words in either direction (i.e. the 10 lowest and 10 highest wordscores)? Report your findings.

Note: Briefly state your choice regarding smoothing for this model. You are not required to implement rescaling. You may assign the anchor texts scores of [-1,1].

(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` versus the true star rating, in the same way as you did for the Hu & Liu dictionaries. Report your findings. By this metric, which did better: dictionaries or wordscores?

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

(a) Briefly explain which pre-processing steps are appropriate for this particular task.

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

(c) In this step, you will train SVM models with a linear kernel. Your goal is to maximize out-of-sample accuracy by fitting models with 5-fold cross-validation.You should fit 3 models, using 20%, 50%, and 70% of the data for cross-validation. The remaining data is the validation set. For example, the last model in this sequence uses 70% of the data for the 5-fold CV, and 30% is saved as the validation set. Report which model has the highest accuracy for out-of-sample predictions made on the validation set. In terms of accuracy, how would you evaluate the performance of this classifier?

(d) 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. Choose the best hyperparameters from the previous question part, and fit an SVM model with those hyperparameters, but with a radial kernel. 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 (75%) and a test set (25%) 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` (set the `importance=TRUE`). After fitting the model, extract the *mean decrease in Gini index* (hint: see `$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 a model parameter. The package's default value for the argument `mtry` is `sqrt(# of features)`. Estimate two more models, one for each of these values of `mtry`: `0.5*sqrt(# of features)` and `1.5*sqrt(# of features)`. As you did above, use each of the fitted models to predict the sentiment values for the test set. Report the respective accuracy scores. Which value of `mtry` yielded the best accuracy?