March 23, 2017
DS-GA 3001-004, Text as Data
Prof Arthur Spirling

Homework 2

This homework must be returned to Patrick Chester at the start of recitation at **7:30 PM on April 4th, 2017**. 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.

You must include an `R` code appendix which is clearly commented, such that it your understanding of the problems can be assessed. You must turn in a paper copy: **no electronic copies will be accepted**.

---

1. We would like you to perform some naive Bayes classification by hand — make sure to show your work!

   (a) Imagine a situation in which you are a French citizen and you receive eight emails - in English for some reason - from the two leading Presidential Candidates: Marine La Pen and Emmanuel Macron. The contents of those emails after all relevant preprocessing are displayed in Table 1. Using the standard Naive Bayes classifier without smoothing, estimate the probability (or rather, the prior multiplied by the likelihood) that the following email was sent from Le Pen or Macron: immigration voter culture help neighborhood. Explain whether you trust your findings and why.

email	content
lepen1	immigration women assimilate afraid win
lepen2	culture voter economy president help
macron1	voter women help reform education
macron2	union economy hope immigration neighbourhood
macron3	win union europe elect president
lepen3	economy voter immigration president culture
macron4	success german culture help french

Table 1: Training set of presidential candidate emails.

(b) Now impose Laplace smoothing on the problem. That is, add one to the numerator and the size of the vocabulary to the denominator, then recalculate your estimates from above. Comment on which candidate is now the more plausible originator of the email.

2. Patrick has gathered 10,000 user movie reviews from Amazon Prime. Each user left a rating of 1-5 along with a written review. Youll be asked to use some of the supervised learning techniques weve discussed in class to analyze these texts.

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

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

   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.

   (a) Before we apply any classification algorithms to the Amazon reviews, we will need a general classifier that tells us whether the review was positive or negative — also referred to as the "actual score". To create it, 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. Also, if it is simpler for you to use numbers rather than labels, you may code "actual score" as 1 if a user score is greater than the median and 0 otherwise.

   - Also, for some tasks, we're interested in having some "anchor" texts at the extreme of the distribution. Create a binary variable ("anchor positive") that is equal to one if the user score given to a review is equal to 5 and is equal to zero otherwise. Next, create a second binary variable ("anchor negative") that is equal to one when the user scores of a given review are equal one; otherwise, "anchor negative" is equal to zero.

(b) The first method we'll use to classify reviews as being positive or negative will be sentiment classification. To do so, you will use the dictionaries of positive and negative words discussed in Hu & Liu (2004) — and available on Patrick's github.

- First, generate a sentiment score for each review based on the number of positive words minus the number of negative words. Then, create a dichotomous vector of equal length in which texts that have a positive sentiment score are considered "positive," while those with a negative score are "negative"; if any of them have a sentiment score of 0, score them as positive. If you used 1 and 0 to code reviews as being positive and negative for "actual score" in 2a), do the same here.

- Create a histogram to visualize the distribution of the continuous sentiment measure. What percentage of reviews have a "positive" sentiment score?

- Determine the usefulness 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 2b1) on one axis and the original binary classifications (created in 2a) on the other axis. Use this confusion matrix to compute the accuracy, precision and recall of the sentiment classifier.

- For both the sentiment score (SentRank) and the actual score (ActRank), generate a rank for that review, where 1 is the most positive review and 10,000 is the most negative. Compute the sum of all of the absolute differences between SentRank and ActRank for each review (see RankSum represented in Equation 1).

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

(c) Now, we'll train a Naive Bayes classifier to predict if a review is positive or negative (according to the original labels).

- Use the "textmodel" function in quanteda to train a *smoothed* Naive Bayes classifier with uniform priors, using 20% of the reviews in the training set and 80% in the test set. What are the accuracy, precision and recall of your predictions?

- Were you to change the priors from "uniform" to "docfreq" would you expect this to change the accuracy of Naive Bayes predictions? Why? Reestimate Naive Bayes with the "docfreq" prior and present the accuracy, precision, and recall of these new results.

- If you try to fit the model without smoothing, it's unable to make predictions about some of the reviews out-of-sample. Why might this be? HINT: think about how these texts differ from the Conservative manfistos we used to train the NB model on in recitation.

(d) 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." And the beautiful thing about this model is that it's simple enough to implement by hand!

- Create a vector of "wordscores" for the words that appear in the "anchor negative" and "anchor positive" reviews using the technique described in Laver, Benoit & Garry (2003). What are the most extreme words?

- Apply these wordscores to the reviews, and 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. Again, compute the absolute value of the sum of all of the differences in rank for each review. By this metric, which did better: dictionaries or wordscores?

(e) Now well 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.

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

- 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–try each value from 10 to 90 (by tens) and report which gives you the highest average accuracy.

- 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?

3. Finally, in the file "CF_rate_trustworthiness.csv", you have been provided the results of a Human Intelligence Task (HIT) conducted on CrowdFlower. The task was for each worker to rate pictures of politicians based on how trustworthy they appeared, from 1 to 10. In the results file, there are a number of variables of interest (credit to the creation of this data goes to Kevin Munger):

   - rating: the rating assigned to the image

- image_name : the name of the image, which contains the race and gender of the person pictured

- _country: the nationality of the worker

The goal of the HIT was to ensure that there was a sample of images of white men, white women and black men that were balanced in terms of their trustworthiness.

- Is there any nationality that is likely to give **statistically significant** — at a 5% level — higher than average ratings?

- Of the three demographic groups in the picture, is there a **statistically significant** difference between the average ratings given to them?

- Likewise, do you observe a **statistically significant** difference in reported trustworthiness that is associated with gender?

- Now, imagine you were working on this project. Describe two questions that you might ask CrowdFlower workers to filter out unreliable coders (i.e. gold standard questions).