DS-GA 1015, Text as Data
Marco Morucci
Assignment date: Feb 28, 2024
Assignment due date: March 14, 2024

Homework 2

This homework must be turned in on NYU Brightspace by **6pm, Mar 14, 2024**. 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 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 `IPython`. **Raw `.py` or `.ipynb` files will not be accepted.** You are responsible for making sure that your homework is fully readable in .pdf format. Anything that is not readable will not be graded.

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 make sure that code lines are not cut (by breaking up any line of code longer than 80 characters).

Part 1

1. (10 pts.) 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) (4pts.) Imagine a situation in which you receive emails from the two main U.S. parties in anticipation of the 2024 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 party: "infrastructure voter growth 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.

email	content
republican1	immigration aliens criminals loophole country
republican2	voter economy tax growth security
republican3	healthcare cost socialism unfair help
democrat1	immigration country growth help voter
democrat2	healthcare inequality expansion unfair economy
democrat3	infrastructure opportunity expansion country security
democrat4	abortion choice right women help

Table 1: Training set of presidential candidate emails.

(b) (2 pts.) Now impose Laplace smoothing on the problem. That is add one to the numerator and the size of the vocabulary to the denominator, then 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).

(c) (2 pts.) Now, suppose a Republican strategist wants to get the following Republican email into a Democrat voter's inbox (they would otherwise be blocked by a *non-smoothed* Naive Bayes filter):

healthcare XXX XXX

From the words available in the Democratic Party emails, which ones should the Republican strategist substitute in for the two place holders (XXX) to *most* increase the probability it makes it into the Democrat's inbox? Make sure to explain your reasoning. *Hint: You would like to find words with the largest difference between the Democrat and Republican posterior probability. Note that the two words can be identical.*

(d) (2 pts.) How would you expect the Democratic voter to alter their email rules in response to this strategy? How does this achieve the Republican strategist's goals? What do we call this general process of degrading filtering via the logic of the Naive Bayes classifier?

Part 2

For this exercise you will use a database of Hotel reviews gathered from Kaggle (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 you use depends on your NYU NetID. If the first number in your NetID is

- 1-3 (e.g., emw128), use `hotelreviews_a.csv`.
- 4-6 (e.g., emw528), use `hotelreviews_b.csv`.
- 7-9 (e.g., emw828), use `hotelreviews_c.csv`.

All data is provided on Brightspace.

Before we get started, be sure to 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.

2. (2pts.) Before we apply any classification algorithms to the Hotel reviews, we will need a general classifier that tells us whether the review was positive or negative—the "true classification."

   (a) (2pts.) 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.

3. (14 pts.) 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.

   (a) (4pts.) First, preprocess the text by cleaning apostrophes, removing punctuation and setting to lower case. Create a `DTM` of the review text. Then, 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.

   (b) (2pts.) Estimate the mean sentiment score in each *true* class together with its standard deviation. Report your results in a table or graph. *You can, but do not have to use bootstrapping for this.*

   (c) (2pts.) 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) (4pts.) 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?)

(e) (2pts.) Now, based on your classification of reviews, inspect the following false positives and false negatives: (1) reviews that received 1 star in reality, but were classified as "positive" using the sentiment score and (2) reviews that received 5 stars in reality, but were classified as "negative" using the sentiment score. Inspect the text of at least 5 of these reviews and comment on why they were misclassified by the sentiment score.

4. (6pts.) Next, we'll train a Naive Bayes classifier to predict if a review is positive or negative. Create a `DTM` with the following preprocessing: remove punctuation, remove numbers, set to lower case, stem terms and remove stop words.

(a) (4pts.) Using scikit-learn, train a 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.). Report the accuracy, precision, recall and F1 score of your predictions. Include the confusion matrix in your answer.

(b) (2pts.) 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. (12 pts.) 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 hotel reviews using the original ordering of the review data. Again, create a `dfm` with the following preprocessing: remove punctuation, remove numbers, set to lower case, stem terms and remove stop words.

(a) (2pts.) Describe an advantage offered by SVM or Naive Bayes relative to the dictionary approach in classifying positive and negative reviews.

(b) (4pts.) 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. Optimize over the relative size of the training and test sets, try each value from 10 to 90 (by 10s). The remaining data is the validation set. In other words, you should train 10 different models with 5-fold cross-validation. For example, the last model in this sequence uses 90% of the data for the 5-fold CV, and 10% 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?

(c) (2pts.) A colleague is building a dictionary for this problem. Using the optimal SVM you identified in 5b), report the five most predictive words for a positive and for a negative review from the SVM results to help them start that process.

(d) (4pts.) Can we improve the performance of our SVM classifier by including bigrams in our model? To evaluate this, create a `dfm` that contains *both* unigrams and bigrams. Make sure to use the same preprocessing as in 5a). Using this data, train another SVM model with a linear kernel and 5-fold cross-validation. You can impose the same relative size of the training and test sets as before (i.e., you do not need to redo 5b). Report the accuracy of this model out-of-sample. How does this compare to your findings in 5b)?

6. (10pts.) For this question use a random sample of 500 reviews in the dataset. 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 using the same preprocessing choices as before (Note: features in the test set should match the set of features in the training set).

(a) (4pts.) Using scikit-learn, fit a SVM model to the training set using the package's default values for `C` (the regularization parameter) and setting `kernel='linear'`. Having fitted the model, extract the *feature weights* (the `coef_` parameter) and order from most important to least important. What are the top 10 most important features according to this measure?

(b) (2pts.) 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.

(c) (2pts.) Now you will do some tuning of one the two model parameters. The package's default value for the argument `C` is 1.0. Estimate two more models, one for each of two different values of `C`: 0.1 and 0.5. 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 `C` yielded the best accuracy? Was there a significant gain in computation speed in changing this feature?

(d) (2pts.) Repeat the exercise above but keeping `C=1`, and comparing model performance when kernel is set to linear, rbf, and poly. Which one yields the best accuracy?