Problem Set 3: Text As Data
IMT 575: Scaling, Applications, and Ethics

Instructor: Lovenoor (Lavi) Aulck
University of Washington
Spring Quarter 2020

April 27, 2020

Instructions

All problem sets must be submitted as a Jupyter notebook. You must organize your analysis and utilize the notebook's markdown capabilities to present your solutions. Partial credit will be awarded for each question for which a serious attempt at finding an answer has been shown. Please see the course syllabus for more instructions/information. A rubric for this assignment is available on the Assignment's canvas page. Unless otherwise noted, this assignment is due on **May 11, 2020.**

Overview

In this problem set, you will perform supervised text classification. In addition to the Jupyter notebook discussed above, you must submit a .txt file with your results. The requirements for this .txt file are described at the end of this document.

The data for the classification task will consist of quotes from candidates vying to be the Democratic presidential nominee for the 2020 US presidential election. All data has been extracted from debates between candidates. You will be tasked with identifying who said what for a subset of unlabeled data. You may not use external data to make predictions. You may not look up the snippets/quotes provided and use that to guide your model design.

You will be provided with two compressed folders: train.zip and test.zip. Each of these contain files with a single quote/snippet in each file from a single speaker. The train.zip folder contains training instances. There are 528 training instances, with the speaker for each identified both in the file name and at the beginning of the text for each file. The test.zip folder contains test instances. There are 111 test instances. No labels will be provided for these instances; it will be your task to generate labels.

There is an opportunity for extra credit as part of this assignment. The extra credit points can be used across all problem sets. Please see the end of this document for details.

Part 1: Set-up

1. To begin, you will first work through processing the data. Start by loading in the training data and test data. Keep each in their own dataframe/list (i.e. a dataframe/list for all training observation text and a separate one for all test observations). How many training and test observations do you have?

2. Create a vector of training labels. These labels can be found in one of two ways: in the file name of each training data observation and at the beginning of the text for each training file. The test data will not have the labels at the beginning of the text so remove these labels from your training observations. Are there any instances where the name of the file does not align with the name at the start of the text? If so, how many such observations are there? Exclude these from your training data. What does the distribution of labels look like? Comment on what you see.

3. Convert the training data to lower case. Remove punctuation from the training data. Also remove stop words from the training data using the NLTK package's English stop word list. In addition to NLTK's stopwords, are there words specific to this dataset that may be worthwhile to treat as stop words? What are these words and why would you exclude them? Remove these additional stop words from the training data as well (note: you can also remove this secondary set of stop words after tokenizing. However, keep in mind that this may cause feature alignment issues with your test dataset when you tokenize it.)

4. Stem/lemmatize your training data using a stemmer/lemmatizer of your choosing. Show a before and after using a few observations and comment on what you see.

5. Tokenize your training data using unigrams (hint: see sklearn's CountVectorizer). If you set upper and lower limits on word frequency, what are they? How many unique tokens are in your vocabulary?

6. Process the test data in a manner identical to the training data. Note that you will need to have the same dimensions for your training and test data. One way in which this can be done is using sklearn's CountVectorizer is to fit on the training data and transform the test data. Show that the number of features for your training and test data are identical.

Part 2: Supervised Learning

7. The primary objective of this problem set is to build classifiers to predict who said what. Having the right features are essential for this (and any) prediction task. In addition to the features you were tasked with generating in Part 1, design/build additional features of your choosing. Please note that this process can be iterative in a cycle of designing/adding features and evaluating model performance - this is completely reasonable and you can come back to this question after completing the remainder of Part 2. Describe the features that you ultimately used in your final models and articulate your reasoning for including the features that you did. Also describe any feature manipulation/scaling you did and your reasoning for doing so. Ensure your test set has identical features as your training set. Some potential ideas for additional features include bi-/trigrams, topic model weights, TF-IDF weights, sentiments, hand-engineered word co-occurrences (not adding any/all of these is completely fine as well).

8. You will create three different models for this problem set: a regularized logistic regression model, a tree-based model (your choice of any tree-based model), and any other model of your choice (regardless of whether it was discussed in class or not). You do not need to implement anything from scratch and can use out-of-the-box models/pipelines. To start, build a regularized logistic regression model. What was your regularization coefficient and how well does this model perform? Are there any classes where it performs particularly well? Are there any classes where it performs particularly poorly? Use graphs/tables where appropriate and contextualize results.

9. Next, build a tree-based model. Which model did you use and why? Are there any classes where it performs particularly well? Are there any classes where it performs particularly poorly? Use graphs/tables where appropriate and contextualize results.

10. Next, build any model other than the ones you've already built. Which model did you use and why? Did you tune hyperparameters? If so, which ones and how? Are there any classes where it performs particularly well? Are there any classes where it performs particularly poorly? Use graphs/tables where appropriate and contextualize results.

11. Compare the performance of the three classifiers you built. Which is the strongest? Which is the weakest? Compare their performance across the different classes and use graphs/tables where appropriate.

12. Using any of the three models of your choosing, generate labels for the test set. Use only last names in all caps as outputs. In other words, ensure every label is one of the following (note the lack of apostrophes and punctuation): BIDEN, BOOKER, BUTTIGIEG, CASTRO, GABBARD, HARRIS, KLOBUCHAR, OROURKE, SANDERS, STEYER, WARREN, YANG. Save these labels as you will include them in your submission, as detailed at the end of this document.

Part 3: Semi-Supervised Learning

13. For this part, we will implement a semi-supervised learning process. As a first step, select which of the classifiers from part 2 you will use. You want a classifier that outputs class probabilities (i.e. `predict_proba` in sklearn). We will be training and re-training this classifier 10 more times so you also want to use a model that does not take long to train. State which model you chose and your thought process for choosing it versus the other models. You may want to copy this model to another variable so you can refer to it in its current form later.

14. The process for semi-supervised learning will involve iteratively adding subsets of our test dataset to our training dataset. To start, using the model you selected in QXX, predict label probabilities on the test set. Take the top 10% of test instances which have the highest probability of having the correct labels (i.e. the highest probabilities of belonging to any class). Add your predicted labels to this 10% and then add it to your training data. Retrain your model on this updated training set. How does the performance on the original training dataset compare between this model and the one you trained in Part 2?

15. Now, repeat the process above 9 more times, each time adding an additional 10% of the test data for which the labels have the highest probability (note that this may result in different test observations being included from one iteration to the next). After each iteration, note the performance on the *original* training data set. Generate a plot which shows the percentage of the test dataset used on the X-axis and the classification accuracy on the original training dataset on the Y-axis. There should be 11 points for this plot, ranging from X = 0% to X = 100%, inclusive. What does the plot look like? Comment on what you see. (Note: this type of semi-supervised learning can very much be hit-or-miss. This type of learning doesn't always yield benefits).

16. Using the model that is trained on the entirety of the training data and 100% of the test data (labeled by you), generate a final set of labels for the test data. Be sure to only use last names

in all caps for the labels, as you did in Question 12. Include these in your submission per the instructions below.

Submission and Extra Credit

In addition to your Jupyter notebook, you must also submit a .txt file. The .txt file should be tab delimited with three columns. The columns should have the following headers (please use all caps): FILE, MODEL1, MODEL2. The first column should contain the name of the file (i.e. test_1.txt, test_2.txt, ... test_111.txt). The second column should contain the labels for the corresponding files from your choice of model from Question 12. The third column should contain the labels for the corresponding files from Question 16.

EXTRA CREDIT: Students with the best-performing classifiers will be given an opportunity for extra credit. This extra credit opportunity will consist of students creating a short (<5 min) video for their classmates detailing their approach to feature generation and model creation. Students will be approached individually about this opportunity after assignments have been graded.

Not Required (Extra Fun)

- NOTE: This question will not be graded and is not required. If you want to carry this analysis further, here's an idea: perform hierarchical clustering on the data to look at which candidates talked about similar things. There are many ways in which this can be done: you can find similarities using LDA or TF-IDF and K-means clustering. From there, you can use something like Scipy's dendrogram functions