CSCI 6350 (Barbosa)
Project 3: Document Classification using Naïve Bayes

Due: See course calendar for due date – may not be turned in late.

Assignment ID: **proj3**
File(s) to be submitted: **proj3.py, myout.log**

Objective(s): Implement a small Python program that makes use of supervised learning classifiers in the Natural Language Toolkit (NLTK). Perform preprocessing and cleaning on the input data.

Project description:
This is a relatively open-ended assignment where your approach counts. In this assignment you will implement a Python program that takes as input 181 State of the Union addresses (92 by Republican and 89 by Democratic U.S. Presidents), and uses a **bag-of-words** approach to extract features and classify each speech as either being made by a Republican or Democratic President. You will hand-pick features, and the cleverness of your method counts.

Requirements:

1. (**60%**) Your program must provide the functionality listed below, in a single file named ***proj3.py***:

- *Input and Preprocessing* – There are two input files in this assignment: 1) ***sotu.txt*** contains the text of 233 State of the Union speeches (including those from Presidents who were neither Republicans or Democrats – these should be excluded and not processed); and 2) ***presidentsbyparty.txt*** lists presidents from the Republican and Democratic parties. You will need to study the *sotu.txt* data file to determine its format, and decide how to extract the information for the presidents of the two parties of interest, and how to format that data for classification. The data was gathered from multiple sources (and thus formats), and some speeches are spread across multiple lines of text. The final format of the data you use as input to the classifier is up to you but the conversion must be included in your code, and must operate directly on the provided input files.
- *Processing* – The processing requirements include:
  - Things to consider:
    - whether or not to remove punctuation like periods, commas, colons, semi-colons, question marks, exclamation points, double quotes, etc.
    - whether to remove stopwords (NLTK stopwords may be augmented by others you believe appropriate for removal)
    - whether to normalize text to lowercase
  - Extract features from the bag-of-words that you believe will be helpful in classifying the speeches. Common bag-of-word features are: counts of (certain) words; presence of (certain) words; (average) length of words; (average) length of sentences; (average) length of (entire) text, or any other word-based feature you can think of.
  - You must have a function that (selects and) loads the features into a data structure for classification. This function should be clearly identified in comments as doing this task. You may use a features dictionary like the one demonstrated in class (from Chapter 6 of the NLTK book), or one of your own.
  - Split your data 80%/20% for training and testing respectively.
- *Output* – Program output:
  - You must use the NLTK Naïve Bayes classifier in this assignment (*nltk.NaiveBayesClassifier*)
  - Your approach should be run 30 times and the results averaged. Report only the final averaged results for overall classifier accuracy; and for precision, recall, and F1 measures on a per-class basis.

- The floor (lowest value you should accept) for your averaged results should be 60% accuracy. This level of performance is easily attainable by selecting some of the features mentioned as suggestions above. Your goal should be to get the best F1 measure possible, without overfitting to the training data.
- The running time of your program must be reasonable, and must take limited computing resources into account. The Naïve Bayes method is fairly robust, and as such there are diminishing returns to using an excessive number of features.
- If your scores are particularly good, you will be asked to share a brief description of your preprocessing and feature selection with the class.

2. (**10%**) Test your program – Re-arrange the data to experiment with different splits for training and test. Randomize you test and training sets.

3. (**5%**) Code comments - Add the following comments to your code:

- A section at the top of the source file(s) with the following identifying information:

  Your Name
  CSCI 6350-001
  Project #X
  Due: mm/dd/yy

- Below your name add comments in each file that gives an overview of the program or class.
- Place a one or two-line comment above each function that summarizes the workings of the function.

4. (**25 %**) Analysis – Provide an answer (as commented code) to the following questions:

  1. Explain what you did to preprocess the data, addressing what you tried that didn't work, and state why you decided on the final approach you chose (include aspects in processing requirements: punctuation, text case, …)
  2. Discuss the features you chose and why: How did you arrive at them? What did you try that did not work?
  3. Use the output of the *show_most_informative_features* method of the classifier (the data is on a per run basis) and generalize why the top 5 features (over all 30 trials) worked as well as they did, given the input data.
  4. What are the possible weaknesses of your approach? Is it likely that independence was violated? Is it possible that there's double counting in your approach? Identify where.

Project submission:

You must complete this assignment using Jupyter Notebook. Submit only the .ipynb file. You may complete this on your home system, or on the CSCI 4850 server (referred to as the NLP system hereon). You **MUST** test it on the NLP system , where it must run without modification. Be mindful that the NLP system does not have the handin utility. You must submit your work from ranger.

Below is an example of how you transfer files between ranger2 and the NLP system:

On NLP System terminal - copy to ranger2: scp test.txt ranger2.cs.mtsu.edu:~/.

On NLP System terminal - copy from ranger2: scp ranger2.cs.mtsu.edu:~/test.txt .

The UP arrow in Jupyter allows the upload of a file on the local machine to where the notebook is.

Once all files are on ranger use your class C account to submit your work. The *handin* command allows assignments to be submitted electronically. The command *handin* must be followed by assignment ID and the files required. The files must be named exactly as listed here (although they may appear in any order). Submit your assignment from the directory where your source and log files are, with the following command (you may resubmit your work – it simply overwrites the previous submission. Your last submission will be graded).

handin proj3 proj3.ipynb