CS534 — Implementation Assignment 2 — Due Oct 18th 11:59PM, 2016

General instructions.

1. The following languages are acceptable: Java, C/C++, Matlab, Python and R.
2. You can work in team of up to 3 people. Each team will only need to submit one copy of the source code and report. Make sure to list the group members, indicate percent project contribution for each member.
4. Your source code and report will be submitted through the TEACH site
`https://secure.engr.oregonstate.edu:8000/teach.php?type=want_auth`
5. The TA will need to be able to run your code. Please provide clear instructions in a text file named "README" on how to compile/run your code and evaluate your model on the dev file.
6. When evaluating the dev file, your code will need to output a separate text file of predictions similar to clintontrump.label.dev (see the section "Data set information" below).
7. Be sure to answer all the questions in your report. You will be graded based on your code as well as the report. In particular, **the clarity and quality of the report will be worth 10 pts**. So please write your report in clear and concise manner. Clearly label your figures, legends, and tables.
8. In your report, the results should always be accompanied by discussions of the results. Do the results follow your expectation? Any surprises? What kind of explanation can you provide?

Naive Bayes (total 70pts)

In this assignment you will implement the Naive Bayes classifier for document classification with both the Bernoulli model and the Multinomial model. For Bernoulli model, a document is described by a set of binary variables, and each variable corresponds to a word in the vocabulary V and represents its presence/absence. The probability of observing a document $\mathbf{x}$ given its class label y is then defined as:

$$p(\mathbf{x}|y) = \prod_{i=1}^{|V|} p_{i|y}^{x_i}(1 - p_{i|y})^{(1-x_i)}$$

where $p_{i|y}$ denotes the probability that the word i will be present for a document of class y. If $x_i = 1$, the contribution of this word to the product will be $p_{i|y}$, otherwise it will be $(1 - p_{i|y})$.

For the Multinomial model, a document is represented by a set of integer-valued variables, and each variable x_i also corresponds to the i-th word in the vocabulary and represents the number of times it appears in the document. The probability of observing a document $\mathbf{x}$ given its class label y is defined as:

$$p(\mathbf{x}|y) = \prod_{i=1}^{|V|} p_{i|y}^{x_i}$$

Here we assume that each word in the document follows a categorical distribution of $|V|$ outcomes and $p_{i|y}$ is the probability that a randomly selected word is word i for a document of class y. Note that $\sum_{i=1}^{|V|} p_{i|y} = 1$ for $y = 0$ and $y = 1$.

Your implementation need to estimate $p(y)$, and $p_{i|y}$ for $i = 1, \cdots, |V|$, and $y = 1, 0$ for both models. For $p(y)$, you can use MLE estimation. For $p_{i|y}$, you MUST use Laplace smoothing for both types of models.

Apply your Naive Bayes classifiers to the provided twitter data. You will learn your models using training data, apply the learned model to predict on the test data[1], and then report the performance.

One useful thing to note is that when calculating the probability of observing a document given its class label, i.e., $p(\mathbf{x}|y)$, it can and will become overly small because it is the product of many probabilities. As a result, you will run into underflow issues. To avoid this problem, you should operate with log of the probabilities.

Data set information: Since it is an election year, we will be providing a more festive politically-related data set. We have provided a set of roughly 6,000 tweets from both Hillary Clinton and Donald Trump's

[1]You will actually be given "dev" or "development" data in lieu of test data. You can treat this as your "test" data to run on. The TA will have a separate test data set to run your programs on when it comes time to grade. This is used to prevent people overfitting on the given dev set.

Twitter accounts[2]. Our task is to classify whether a given tweet came from Hillary or The Donald. We have provided four files:

- clintontrump.tweets.train is a text file containing a tweet on each line. The text has been cleaned up slightly in that it has been tokenized and some odd unicode has been stripped out.
- clintontrump.label.train is a text file that will say Hillary Clinton or Donald Trump on each corresponding line.
- clintontrump.tweets.dev is similar to the train file, containing a tweet on each line. Use this file to test your model.
- clintontrump.label.dev is the corresponding text file with the labels. Use this file to check if your model predicted correctly on the dev set. Your code should output a separate file that looks like this one, but with a prediction on each line. The TA will use that separate file to evaluate the accuracy of the model.

Basic implementation:

1. (5 pts) Please explain how you use the log of probability to perform classification.
2. (5 pts) Report the overall testing accuracy (number of correctly classified documents over the total number of documents) for both (Bernoulli and Multinomial) models.
3. (5 pts) Whose tweets were confused more often than the other? Why do you think this is? To answer this question, you might want to produce a K by K confusion matrix, where $K = 2$ is the number of classes (Clinton, Trump), and the i, j-th entry of the matrix shows the number of class i documents being predicted to belong to class j. A perfect prediction will have only diagonal elements in this confusion matrix.
4. (5 pts) Identify, for each class, the top ten words that have the highest probability.

Priors and overfitting:
(20 pts) In this part, we will focus on the multinomial model and experiment with different priors. In particular, your last set of experiments use' Laplace smoothing for MAP estimation, which corresponding to using $Dirichlet(1+\alpha, ..., 1+\alpha)$ with $\alpha = 1$ as the prior. This can be viewed as including a total of $|V| * \alpha$ "fake" samples, with α (α must positive, but can be non-integer) samples per word. In this part, you will retrain your classifier with different values of α between 10^{-5} and 1 and report the accuracy on the test set for different α values. Create a plot with value of α on the x-axis and test set accuracy on the y-axis. Use a logarithmic scale for the x-axis. Comment on how the test set accuracy change as α changes and provide a brief explanation for your observation.

Identifying important features:
(20 pts) For this part, design and test a heuristic to reduce the vocabulary size and improve the classification performance. This is intended to be open-ended exploration. Please describe clearly what is your strategy for reducing the vocabulary size and the results of your exploration. A basic pointer to seed your exploration is that we would like to remove words of no discriminative power. How can we measure the discriminative power of a word?

Bonus:
(Bonus: 10 pts) What else could you do to improve the performance of your classifier? Here are some suggestions to help you get started.

- Can you change the smoothing?
  - Can you change how you smooth over the counts of the words you find in the training data?

[2]The original data set was downloaded from: `https://www.kaggle.com/benhamner/clinton-trump-tweets`. The files we are providing were further processed from what they give you.

  - Can you change the way you handle unknown words in your dev set? [3]

- Can you add additional features?
  - What if you tried looking at all combinations of two words in succession? Three words? More?
  - What about particular phrases ("I'm with her", "Make America Great Again")
  - Noah Smith has a Twitter Parser (`http://www.cs.cmu.edu/~ark/TweetNLP/`). Maybe you could try running it on some tweets to see what you get? For example, if you added all the Part of Speech tags as features, would you get an improvement?

- Do some error analysis. Take a look at the specific tweets you happen to be misclassifying. Is there something you could do to improve your model in handling those (without degrading the performance of your model elsewhere)? Or are these tweets just too noisy to do anything with?

Mention what you tried at the bottom of your report. Bonus credits will be given to the groups with the top accuracy performance on his separate test set.

[3]Check out Kneser-Ney Smoothing