Erick Suarez
4292702
erick_suarez@ucsb.edu

• Architecture: Explanation of your code architecture (briefly describe the classes and basic functionality):

I have a class NaiveBayesClassifier that has the following public methods:
trainWithDataSet(...): parses data and fills up dictionaries that contain word frequencies for each document

createModel(...): uses the data from trainWIthDataSet to fill in dictionaries with data that will be used later on such as P(Author) and conditional Probabilities P(word|Author)

classifyDataSet(...): classify data set parses the file and for every document and for every author it classifies the document based off the top 33% most significant words using TF-IDF to calculate the significance of each word and looks up the conditional probabilities of said words, multiplying all the probabilities and finishing off by multiplying the total with P(Author). All the while keeping track of the classification with the highest probability which it outputs in the end.

• Preprocessing: How you cleaned and how you represent a document (features)
I used the bag of words model, therefore no cleaning and my features were the words themselves.

• Model Building: How you train the Naive Bayes Classifier
Because I used the bag of words model, I just calculated the conditional probabilities (using the naive bayes smoothing formula) for all words given all authors as well as the probability that a document belongs to an author and use those values to calculate P(Author | x1,x2…) .

• Results: Your results on the provided datasets (accuracy, running time). Also give a sample of the 10 most important features for each class (positive or negative)
4.3053319454193115 seconds (training)
13.83262300491333 seconds (labeling)
0.969 (training)
0.13766666666666666 (testing)

Because I used the bag of words model the 10 most important features for each class would be the words with the largest conditional probabilities given said class.

• Challenges: The challenges you faced and how you solved them

The Challenges I faced for the first half of the project was a dictionary bug for a class field that is a dictionary with key being a word and value being array from 0 to number of authors, and when I would update an index in one of the arrays it would change the values for other keys in the same index to the same value. This originated to how I was populating the empty map, because I was using the same default array for all the map values instead of creating a new one for each key.

After that the last challenge I faced and defeated me was improving the testing accuracy. I added more features, such as bigrams, and trigrams, using stop words, using tf-idf to only calculate the cond probabilities of the most import terms to the document being classified for every author, but either the accuracy wouldn't budge or it would decrease. Ultimately I grew very tired and frustrated and decided to just turn it in.

• Weaknesses: Weaknesses in your method (you cannot say the method is without flaws) and suggest ways to overcome them.

- Words that were spaced out such as : su ch (such) were treated as separate words despite it being intended to be one word
- Overfitting
- Semantic relationship between words isn't taken into account