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 3 main methods:
trainWithDataSet(...): parses file and for each document fills up dictionaries that contain word frequencies pertaining to that 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(...): parses the file and for every document it calculates the probability of it belonging to each author and returns the author with the highest probability.

• Preprocessing: How you cleaned and how you represent a document (features)
I cleaned the documents by removing some stopWords. My features were all the words that appear in the document that's being classified.

• 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 in a given document multiplied by the probability of the given author (number of documents written by author/total number of documents). I did this for all authors given the document.

• 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)
2.885037899017334 seconds (training)
13.275662899017334 seconds (labeling)
0.9846666666666667 (training)
0.15033333333333335 (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, b/c training is very accurate but testing isn't, and was confused on this because I thought the naive bayes smoothing formula took care of this.
- Semantic relationship between words isn't taken into account