Programming for Data Analytics Project B

Bayesian Classification

The objective of this project is to build a Naïve Bayes classifier that predicts the sentiment of tweets.

Project Specification.

Naive Bayes classifiers are among the most successful known algorithms for learning to classify text documents. The primary technical objective of this exercise is to implement a **Multinomial Naive Bayes** learning algorithm in Python to classify tweets.

In the same folder as the lab exercise you will find a zip file called dataFiles.zip. The zip file contains a train and test folder. The train folder contains two separate data files **trainNeg.txt** (400, 000 lines where each line is a tweet that has been classified as negative) and **trainPos.txt** (400, 000 lines where each line is a tweet that has been classified as positive).

Once you have trained your model you should assess the accuracy of your model using the files contained in the test folder.

Naïve Bayes will treat the presence of each word as a single feature/attribute. This would give you as many features as there are words in your vocabulary. The Multinomial model places emphasis on the frequency of occurrence of a word within documents of a class (see lecture slides for more details and examples).

Stage 1 –Vocabulary Composition and Word Frequency Calculations

Develop code for reading all tweets **from both the positive and negative files**.

You should initially create a data structure to store all unique words in a vocabulary. A **set** data structure in Python is appropriate for this purpose. You can keep adding lists of words to the set and it will only retain unique words.

Your next step is to record the frequency with which words occur in both the positive and negative tweets. I recommend that you use dictionaries to store the frequency of each word. (Note the keys of each dictionary should correspond to all words in the vocabulary and the values should specify how often they occur for that class). For example, if the word “brilliant” occurs 55 times in the positive tweets then the key value pair in your positive dictionary should be <”brilliant” : 55>.

It can be useful when initially creating the positive or negative dictionary to use the values from the set (which contains all your unique words) to initialize all the keys for the dictionary. See example code below:

```
# this line creates a dictionary, which is initialized so that
#each key is a value from the set data structure vocab

negDict = dict.fromkeys(vocab, 0)
```

Note:
When opening the individual training and test files you can use the following code:
dataFile = open("trainPos.txt","r")
Depending on your environment this may raise an encoding related error.
An alternative is to read in to the file using the following lines.
import codecs
dataFile = codecs.open("trainPos.txt", 'r', encoding = "ISO-8859-1")

Stage 2 – Calculating Word Probability Calculations

Once you have populated your positive and negative dictionary with the frequency of each word, you must then work out the conditional probabilities for all words (for each class). In other words for each word *w* you should work out the *P(w|positive)* and *P(w|negative).* Refer to lecture notes (Bayesian Classification) for more information. Remember this is a multinomial model.

Stage 3 – Classifying Unseen Tweets and Performing Basic Evaluation

The final section of your code will take as input a new tweet (a tweet that has not been used for training the algorithm) and classify the tweet as a positive or negative review. You will need to read all words from the tweet and determine the probability of that tweet being positive and the probability of it being negative.

For the basic evaluation of your algorithm you should run all tweets from the test folder through your algorithm and determine the level of accuracy (the percentage of tweets correctly classified for each class).

Classification on twitter is challenging but you should expect to see an average accuracy in the region of 75%.

Stage 4 - Additional Elements

This section is for the investigations of the impact of some pre-processing techniques on the accuracy. Common techniques include lowering the case of all words, punctuations removal, stop-word removal, n-grams, etc.

The regular expression library in Python may prove useful in performing pre-processing techniques. (re module https://docs.python.org/3.6/library/re.html). This provides capabilities for extracting

whole words and removing punctuation. See example on the next page. You can find a tutorial on regular expression at https://developers.google.com/edu/python/regular-expressions .

An alternative is the use of NLTK, Pythons natural language toolkit (http://nltk.org/). Note to use this from Spyder you will need to run nltk.download('all'). It is a power library that provides a range of capabilities including stemming, lemmatization, etc.

Stage 5 - Visualization

You need to visualize the classification results before and after the pre-processing using two-dimensional graphs.
You can use **matplotlib** and **seaborn** Python libraries.

Optional: Steaming and Classifying Tweets from Twitter

For those of you that might be interested it is possible to create a developer account with Twitter. You can use the Twitter API to stream tweets based on a specific keyword. Go to the following for a guide.
https://pythonprogramming.net/twitter-api-streaming-tweets-python-tutorial/
Please note it can take a little bit of time to get this working. You can pipe these tweets through your model in order to classify sentiment. For example, you could increase the probability threshold for both positive and negative and classify sporting events and monitor/graph sentiment throughout the event.

Guidelines and Submission Instructions:

1. The project is worth 50% of your overall module grade. You will produce a **.py file** containing all your code and a **short document** illustrating the sample output for each of the operations outlined the project.

2. Upload your solution python file (**.py file**) to Blackboard before **13:00 on Friday Dec 14th**.

3. Go to the Project B folder in Blackboard to upload your file.

4. Once you have submitted your files you should verify that you have correctly uploaded them. It is your responsibility to make sure you upload the correct files.

5. Please make sure you **fully comment your code**. You should clearly be explain the operation of important lines of code.

6. Please put your student name and number as comments at the top of your file.