Machine Learning – Bayesian Classification

Due Date:

Assignment should be submitted to Backboard before 10pm **Sunday Oct 22nd .**

Assignment Marks:

The distribution of marks is as follows:

1. **Naive Bayes Algorithm 60%**
2. **A Basic Evaluation 10%**
3. **Research and Detailed Evaluation 30%**

Parts 2 and 3 above will be submitted as a report along with your Python code from part 1.

The report should consist of two parts:

(i) The basic evaluation should describe the basic methods you have employed for cleaning the dataset (for example, converting everything to lower-case, removal of punctuation, etc). It should also provide an account of the performance of the model and how (or if) it was impacted by the basic methods of cleaning the data.

(ii) The research and detailed evaluation of the algorithm should investigate the impact of more advanced pre-processing techniques on the classification accuracy of your Naïve Bayes classifier. Remember you should test your algorithm using data that was not used to train the algorithm in the first place. The research element allows you to explore and report on various efforts you have made to improve the classification accuracy of the algorithm. Examples, of more advanced techniques would be stemming, methods for dealing with negation, n-grams, etc

Objectives

Naive Bayes classifiers are among the most successful known algorithms for learning to classify text documents. The primary technical objective of this assignment is to provide an implementation of **a Multinomial Naive Bayes** learning algorithm in Python for document classification.

The dataset you will be using consists of movie reviews from IMDB (Internet Movie Database). It contains two folders. One called ***neg***, which contains negative movie reviews and the other called

pos which contains positive movie reviews. The objective of your Naïve Bayes algorithm is to correctly classify unseen movie reviews as positive or negative.

Please note I have provided two variants of the IMDB datasets: (1) IMDB_Small and (2) IMDB_Large. The small dataset contains 1000 negative and 1000 positive reviews. The large dataset contains 25,000 movie reviews split evenly between positive and negative. I recommend you use the large dataset but please be aware that depending on the spec of your machine you may find that your algorithm takes a long time to run using the large dataset. If you encounter these problems please let me know. You can use the small dataset for developing your code but your evaluation of the algorithm should be based on the larger dataset. The large dataset is cleaner and will provide more consistent results.

You will also notice the data folder contains a folder called **training**, which you can use to test your algorithm. The reviews contained in training are unseen movie reviews that have not been used to train the algorithm.

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. You should use a "bag of words" (Multinomial model) approach. The Multinomial model places emphasis on the frequency of occurrence of a word within documents of a class (See Week 4 lecture slides for more details and an example).

Stage 1 – File Parsing and Vocabulary Composition

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

```
import os

path = 'SmallIMDB\\neg\\'

listing = os.listdir(path)

for eachFile in listing:

   print "current file is: " + eachFile

   f = open(path+eachFile, "r",  , encoding="utf8")
```

You should initially create a data structure to store all unique words in a vocabulary. A **set** data structure in Python is ideal 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 movie reviews. I suggest that you create a dictionary to store the frequency of words in the negative reviews and another dictionary to store the frequency of words in the positive reviews. (The keys to 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 movie reviews then the key value pair in your positive dictionary should be <”brilliant” : 55>. You need to record the frequency of all the words for each class (positive and negative).

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:

```
vocab = set(["this", "is", "an", "example"])

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

negDict = dict.fromkeys(vocab, 0)

print (negDict)

# prints the following

# {'is': 0, 'example': 0, 'an': 0, 'this': 0}
```

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).* These are the required probabilistic components of the Naïve Bayes model. I suggest that you create a dictionary for the positive conditional probabilities and another for the negative conditional probabilities.

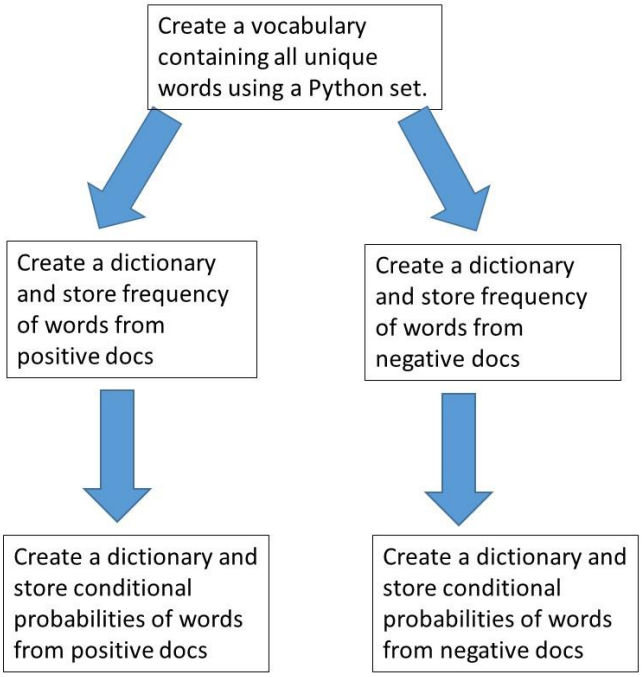
Remember you are building a multinomial model therefore you should be using the following formula for calculating the conditional probabilities.

$$P(w \mid c) = \frac{count(w,c)+1}{count(c)+|V|}$$

count(w, c) is the number of occurrences of the word w in all documents of class c.

count(c) The total number of words in all documents of class c (including duplicates).

|V| The number of words in the vocabulary



Stage 3 – Classifying Unseen Documents and Basic Evaluation

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

Remember to use the log in your probability calculations. See lecture notes for more details on probability classification.

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

You should also try to clean the dataset by lower-casing all words and removing punctuation as much as possible. Your basic evaluation should describe the basic steps you took and if any impact on accuracy was observed.

Research and Detailed Evaluation

The research aspect of this project is worth 30%. You should research common methods used for potentially improving the classification accuracy of your Naïve Bayes algorithm. Please note that basic techniques such as lowering the case of all words and punctuations removal will not be

considered. Your report should provide a detailed account of the research, the subsequent implementation as well as the updated results. You should cite all sources you used. Please note that you will not be docked marks for techniques that do you improve accuracy. It is quite difficult to improve the initial level of accuracy with this dataset. Normally with this dataset you should be expecting to obtain initial accuracy values between 75-80%. However, it is possible to push the accuracy up to 90%.

You can commence this process by employing pre-processing techniques such as those outlined in the lecture slides (stopword removal, stemming, 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.