

## Programming Assignment 4: Naïve Bayes

### Instructions:

- The aim of this assignment is to give you an initial hands-on regarding real-life machine learning application.
- Use separate training and testing data as discussed in class.
- You can only use Python programming language and Jupyter Notebook.
- There are two parts of this assignment. In part 1, you can only use **numpy**, **scipy**, **pandas**, **matplotlib** and are **not allowed** to use **NLTK**, **scikit-learn** or **any other machine learning toolkit**. However, you have to use **scikit-learn** in part 2.
- **Carefully read the submission instructions, plagiarism and late days policy at the end of assignment.**
- Deadline to submit this assignment is: **Sunday 29<sup>th</sup> November, 2020.**

### Problem:

The purpose of this assignment is to get you familiar with sentiment classification. By the end of this assignment you will have your very own “Sentiment Analyzer”. You are given with [Large Movie Review Dataset](#) that contains separate labelled train and test set. Your task is to train a Naïve Bayes classifier on train set and report accuracy on test set.

### Dataset:

The core dataset contains 50,000 reviews split evenly into 25k train and 25k test sets. The overall distribution of labels is balanced (25k pos and 25k neg). There are two top-level directories [train/, test/] corresponding to the training and test sets. Each contains [pos/, neg/] directories for the reviews with binary labels positive and negative. Within these directories, reviews are stored in text files named following the convention [[id]\_[rating].txt] where [id] is a unique id and [rating] is the star rating for that review on a 1-10 scale. For example, the file [test/pos/200\_8.txt] is the text for a positive-labeled test set example with unique id 200 and star rating 8/10 from IMDb. The dataset can be [downloaded from here](#).

### Preprocessing:

In the preprocessing step you’re required to remove the stop words and punctuation marks and other unwanted characters from the reviews and convert them to lower case. You may find the [string](#) and [regex](#) module useful for this purpose. A stop word list is provided with the dataset.

### Part 1:

Implement Naïve Bayes for sentiment analysis from scratch keeping in view all the discussions from the class lectures. Feel free to read [Chapter 4 \(Section 4.1, 4.2, 4.3\)](#) of [Speech and Language Processing](#) book to get in-depth insight of the Naïve Bayes classifier.

Use Bag-of-words representation and apply Laplace (Add-1) smoothing as discussed in class lectures. Specifically, you will need to implement the following algorithm:

```

function TRAIN NAIVE BAYES(D, C) returns  $\log P(c)$  and  $\log P(w|c)$ 

for each class  $c \in C$            # Calculate  $P(c)$  terms
     $N_{doc}$  = number of documents in D
     $N_c$  = number of documents from D in class c
     $\text{logprior}[c] \leftarrow \log \frac{N_c}{N_{doc}}$ 
     $V \leftarrow$  vocabulary of D
     $\text{bigdoc}[c] \leftarrow \text{append}(d)$  for  $d \in D$  with class c
    for each word  $w$  in V           # Calculate  $P(w|c)$  terms
         $\text{count}(w, c) \leftarrow$  # of occurrences of  $w$  in  $\text{bigdoc}[c]$ 
         $\text{loglikelihood}[w, c] \leftarrow \log \frac{\text{count}(w, c) + 1}{\sum_{w' \in V} (\text{count}(w', c) + 1)}$ 
    return  $\text{logprior}, \text{loglikelihood}, V$ 

function TEST NAIVE BAYES( $\text{testdoc}, \text{logprior}, \text{loglikelihood}, C, V$ ) returns best c

for each class  $c \in C$ 
     $\text{sum}[c] \leftarrow \text{logprior}[c]$ 
    for each position  $i$  in  $\text{testdoc}$ 
         $\text{word} \leftarrow \text{testdoc}[i]$ 
        if  $\text{word} \in V$ 
             $\text{sum}[c] \leftarrow \text{sum}[c] + \text{loglikelihood}[\text{word}, c]$ 
    return  $\text{argmax}_c \text{sum}[c]$ 

```

The expected accuracy on test set is around 80%. Use the procedural programming style and comment your code thoroughly.

## Part 2:

Use scikit-learn's [CountVectorizer](#) to transform your train and test set to bag-of-words representation and [Naïve Bayes](#) implementation to train and test the Naïve Bayes on the provided dataset. Use scikit-learn's [accuracy\\_score](#) function to calculate the accuracy and [confusion\\_matrix](#) function to calculate confusion matrix on test set.

## Submission Instructions:

Submit your code both as notebook file (.ipynb) and python script (.py) on LMS. The name of both files should be your roll number. If you don't know how to save .ipynb as .py [see this](#). **Failing to submit any one of them will result in the reduction of marks.**

## Plagiarism Policy:

The code MUST be done independently. Any plagiarism or cheating of work from others or the internet will be immediately referred to the DC. If you are confused about what constitutes plagiarism, it is YOUR responsibility to consult with the instructor or the TA in a

timely manner. No “after the fact” negotiations will be possible. The only way to guarantee that you do not lose marks is “DO NOT LOOK AT ANYONE ELSE'S CODE NOR DISCUSS IT WITH THEM”.

### **Late Days Policy:**

The deadline of the assignment is final. However, in order to accommodate all the 11th hour issues there is a late submission policy i.e. you can submit your assignment within 3 days after the deadline with 25% deduction each day.