CMPE 493

INTRODUCTION TO INFORMATION RETRIEVAL

PROJECT 2 - NAÏVE BAYES

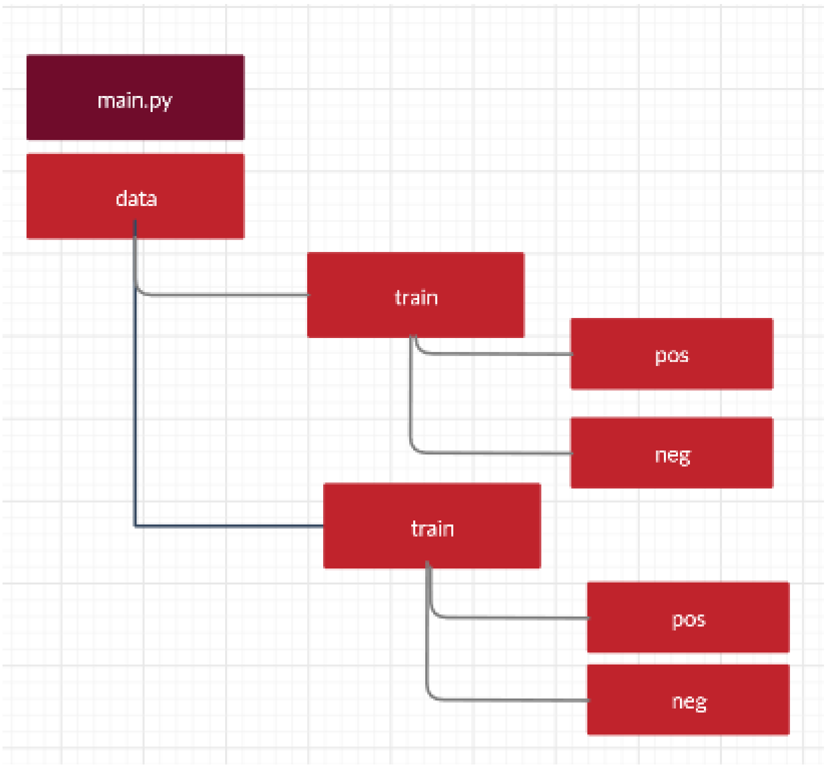
ABDULLAH YILDIZ

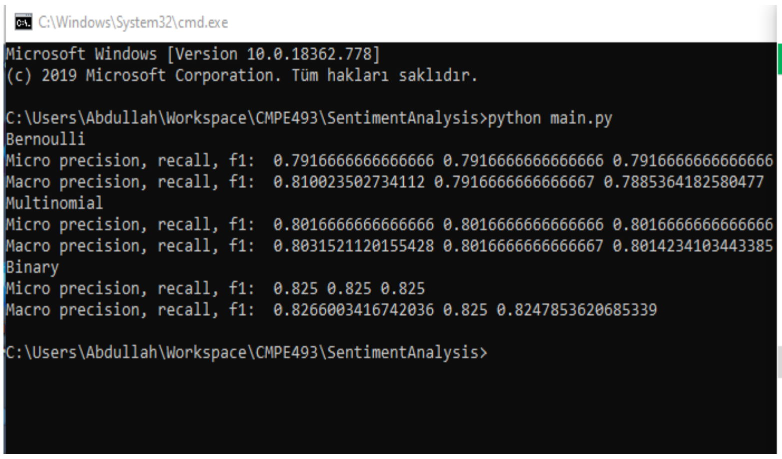
2018400291

How to Run

My project is written in Python.

Make sure data files are under data folder. Specifically, workspace should look like:

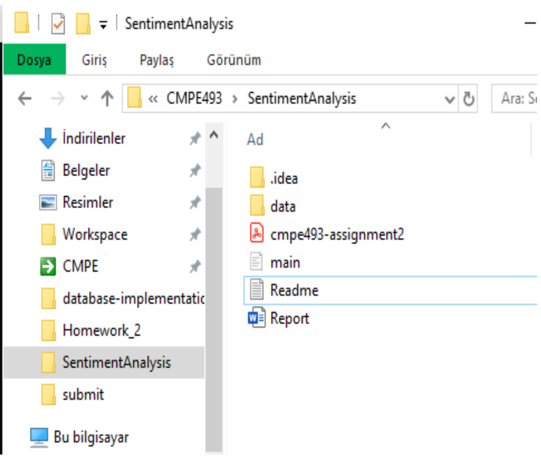




```
Microsoft Windows [Version 10.0.18362.778]
(c) 2019 Microsoft Corporation. Tüm hakları saklıdır.

C:\Users\Abdullah\Workspace\CMPE493\SentimentAnalysis>python main.py
Bernoulli
Micro precision, recall, f1:  0.7916666666666666 0.7916666666666666 0.7916666666666666
Macro precision, recall, f1:  0.810023502734112 0.7916666666666667 0.7885364182580477
Multinomial
Micro precision, recall, f1:  0.8016666666666666 0.8016666666666666 0.8016666666666666
Macro precision, recall, f1:  0.8031521120155428 0.8016666666666667 0.8014234103443385
Binary
Micro precision, recall, f1:  0.825 0.825 0.825
Macro precision, recall, f1:  0.8266003416742036 0.825 0.8247853620685339

C:\Users\Abdullah\Workspace\CMPE493\SentimentAnalysis>
```



Introduction

In this project, my aim is to apply Naïve Bayes models for sentiment analysis task. The dataset that I used is Cornell Movie Review Data Set which has 700 positive and 700 negative movie review for training and 300 positive and 300 negative movie review for testing. I am going to apply 3 different Naïve Bayes method namely Multinomial Naïve Bayes, Bernoulli Naïve Bayes and Binary Naïve Bayes. Then, I will discuss their performances in terms of their micro and macro averaged precisions, recalls and F1-measures.

Algorithms:

1. Multinomial Naïve Bayes

   Multinomial Naïve Bayes is a probabilistic language model that calculates the probability of a document d in class c.

Probability is given by :

$$P\left(\frac{c}{d}\right) \alpha \ f\left(\frac{d}{c}\right) * P(c)$$

In this formula, $f\left(\frac{d}{c}\right)$ *is likelihood. d is a set of words. Likelihood can be calculated as* :

$$\prod_k P(xk|c)$$

k is the positions of the words in document d. But, we use in this project Bag of Words assumption, so positions of words do not matter. We can comment on likelihood that each xk contributes to probability of being in class c.

P(c) is the class frequency that means the proportion of documents in class c out of all documents.

$$p(c) = \frac{Nc}{N}$$

In this algorithm, we try to maximize the posterior that will give us a clue on the class of document d.

Probability is given by:

C = $argmax(P(c|d) = P(c) * \prod_k P(xk|c)$

Now, there is a situation calculating the likelihood. What if there is a probability 0 of a word in class c. Then, it will make whole result zero. To overcome this, we use a little trick and apply a logarithmic transform to probabilities. Since logarithm is non-decreasing function it does not change the characteristics of the algorithm.

$$Likelihood = \log\left(\prod_k \left(P(xk|c)\right) = \sum \log\left(p(xk|c)\right)\right.$$

In my project, I have first formed a large text by concatenating all documents for each class. Then, do a tokenization. Then, for each token in the corpus, I counted their number of occurrence. I used these counters for testing.

Testing:

Given a document d, I tokenized the document and collected under tokens variable. For each token, I added logarithm of its probability and get a result.

2. Bernoulli Naïve Bayes

Bernoulli Naïve Bayes algorithm is interested in whether a word is present in a document or not. The number of times it is contained in a document d is meaningless. So, our results differ from multinomial NB.

Training

To train a Bernoulli NB model, I first created a dictionary bernoulli_probs_P

And bernoulli_probs_N. bernoulli_probs_P holds the probability of documents a word passes.

For example: out of 100 documents if bernoulli_probs_P['the'] is 0.98 then 'the' word is contained in 98 of the positive documents.

I have stored these probabilities for each word in the vocabulary.

Testing

Testing this model starts with tokenizing the document. Then I iterate over vocabulary and get the probability of occurrence in each class with bernoulli_probs_P and bernoulli_probs_N.

Then, if the word is one of tokens in the test set, I add the probability or I add (1 - probability) to all summation of words.

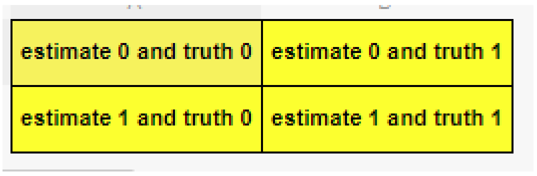
I repeat this for positive and negative sets, whichever summation is bigger wins and class estimation is determined.

3. Binary Naïve Bayes

Binary NB is essentially a Multinomial NB with a difference of not taking into account of the tokens that pass multiple times in a document. My structure for Binary and Multinomial is 99% same. Only difference is I iterate over set of tokens in the Binary NB.

EVALUATION :

In evaluating my test set, I have used a 2x2 matrix and filled it with the estimate and truth pairs.



estimate 0 and truth 0	estimate 0 and truth 1
estimate 1 and truth 0	estimate 1 and truth 1

First row shows the numbers my algorithm decided as negative.

Second row shows the numbers my algorithm decided as positive.

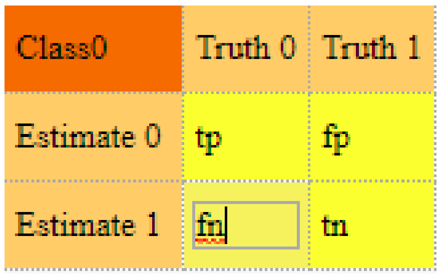
First column shows the numbers that are actually negative.

First column shows the numbers that are actually positive.

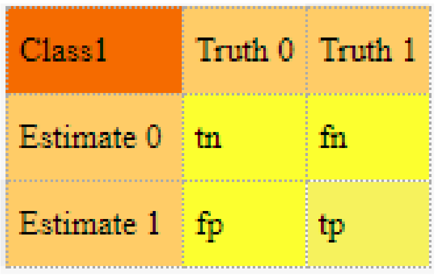
To calculate the true positive, false positives.. I needed to derive two other tables from the matrix above in other perspectives.

e.g. Positive class data gets true positives from 1,1 element of matrix .

e.g. Negative class data gets true positives from 0,0 element of matrix .

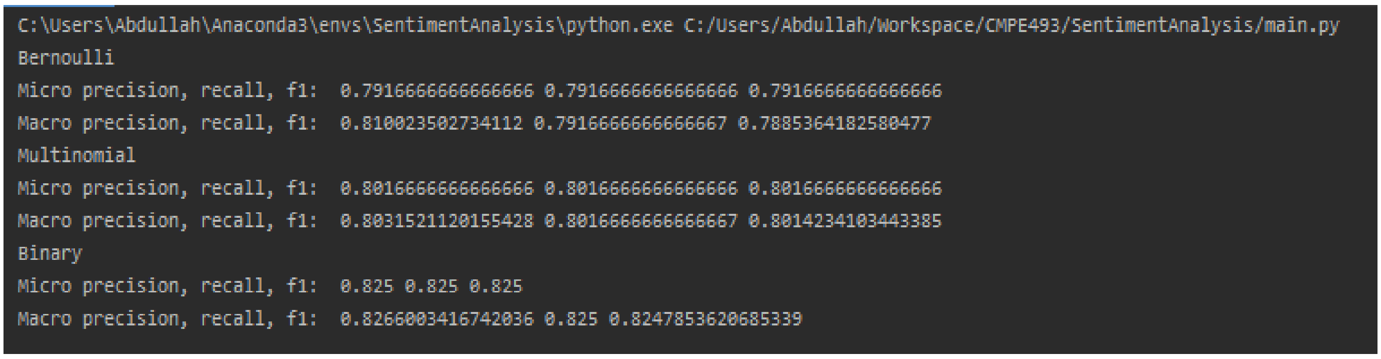


Class0	Truth 0	Truth 1
Estimate 0	tp	fp
Estimate 1	fn	tn



Class1	Truth 0	Truth 1
Estimate 0	tn	fn
Estimate 1	fp	tp

The results for 3 different naïve bayes algorithm are shown below in figure. General results are around 80% which is I believe good numbers.



```
C:\Users\Abdullah\Anaconda3\envs\SentimentAnalysis\python.exe C:/Users/Abdullah/Workspace/CMPE493/SentimentAnalysis/main.py
Bernoulli
Micro precision, recall, f1:  0.7916666666666666 0.7916666666666666 0.7916666666666666
Macro precision, recall, f1:  0.810023502734112 0.7916666666666667 0.7885364182580477
Multinomial
Micro precision, recall, f1:  0.8016666666666666 0.8016666666666666 0.8016666666666666
Macro precision, recall, f1:  0.8031521120155428 0.8016666666666667 0.8014234103443385
Binary
Micro precision, recall, f1:  0.825 0.825 0.825
Macro precision, recall, f1:  0.8266003416742036 0.825 0.8247853620685339
```

- From the results we see that Binary NB algorithm is better option for our data set. The reason it is the best algorithm might be that it oversees multiple occurrence of words in a document. So, what only matters is that is test document full of words that symbolize positive meaning or negative ? The data set is probably more suitable for Binary NB.
  Although the results are close to each other, Bernoulli NB falls behind of all others. The reason behind its low performance might be that it takes into account of the number of documents each word is contained in documents. Also, it checks for each word in vocabulary which is very large and many words are not in the test document. So, this results in negative points for class predictions.

- Macro > Micro

Micro averaging is done by adding tp, tn, fp, fn results of each class and calculate the precision, recall and f1measure from the summed values. However, macro averaged results are simply mean of measures of each class. We see from results that macro averaged results give slightly more accurate estimates.

- Precision , Recall

In Micro averaged measures, precision and recall measures are same but Macro averaged measures are different and precision is better than recall. I can say 3 algorithms return more relevant results than irrelevant ones instead of returning most relevant results. .