MATH 4824B, PROJECT 2

BASIC INFORMATION-

NAME: GUPTA, AYUSH

STUDENT ID: 20384050

FINAL RANK: 60

DISPLAY NAME (ON LEADERBOARD): Ayush GUPTA (20384050)

KAGGLE PROFILE NAME: 20384050

SCORE: 0.59919, ABOVE BASELINE 80

ALGORITHMS: NAÏVE-BAYES ALGORITHM

PARAMETER TUNING: By splitting the training dataset into two parts of 13000 and 3000 entries, then training the model on 13000 entries and then testing on the 3000 entries, checking the accuracy of the model and tuning the parameter (alpha) of Laplace smoothing. Initially, I used alpha as 1, then experimented with 0.5, 2, 0.6, 0.4, 0.1. This gave me a good range (0.3 to 0.7, best at 0.5) of alpha, for which our model works best on this project. Later I used the entire dataset for training and tested it on the available test data, then submitted the files on Kaggle competition, then based on the accuracy result, I tried different values of alpha. It seems that alpha = 0.44 works best here, giving accuracy of 0.59919. [alpha = 0.45 gives accuracy of 0.59875]

HOW TO RUN MY CODE: Just run the 20384050_CODE_PROJ2.py file. Datasets are already in folder data.

THIRD PARTY LIBRARIES: See the following code-

```
import numpy as np

from scipy import sparse

from collections import Counter

from sklearn.metrics import accuracy_score, precision_recall_fscore_support

import nltk

from nltk.corpus import stopwords

import string

import pandas as pd

from nltk.stem.wordnet import WordNetLemmatizer

from nltk.tokenize import ToktokTokenizer

lemma = WordNetLemmatizer()

token = ToktokTokenizer()
```

DETAILED INFORMATION OF THE PROJECT:

Part 1: More than parameter tuning:

Lemmatization:

Meaning of Lemmatization -

“The goal of both stemming and lemmatization is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form. For instance:

am, are, is ⇒ be
car, cars, car's, cars' ⇒ car

The result of this mapping of text will be something like:

the boy's cars are different colors ⇒
the boy car be differ color”

https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html

How I applied lemmatization in this project –

In lines 9,10,11,12 code to call functions:

“ from nltk.stem.wordnet import WordNetLemmatizer

from nltk.tokenize import ToktokTokenizer

lemma = WordNetLemmatizer()

token = ToktokTokenizer() “

In line 33: “ x = lemma.lemmatize(word, 'v') “

This code makes sure that every token is lemmatized before being added to the tokens list.

Stopwords ‘no’ and ‘not’ can be useful:

In lines 15,16,17:

stop_words = set(stopwords.words('english') + list(string.punctuation))

stop_words.remove('no')

stop_words.remove('not')

I removed the words ‘no’ and ‘not’ from the stopwords list, because these words can have importance in sentiment analysis. [Though they will work best when n-gram analysis is used. In this project, I did not use n-gram analysis]

In this way the accuracy of Naïve Bayes model for our project is increased to some extent. The scores of using alpha as 0.5 on model (lab 4 code) without the part 1 is 0.59652 and with part 1 is 0.59697. That’s

some improvement (from 59.652% to 59.697%), but not a very big improvement. It is still considerable improvement because alpha = 0.5 might not be the best when part 1 is applied.

Part 2: Splitting the training data into 13000 and 3000 entries: Parameter tuning

I split the training data into training data 1 (tr1) and training data 2 (tr2). Then, I trained our modified Naïve Bayes model on tr1, and checked the accuracy for tr1 and tr2. By trying different values of alpha (line 84,85,86) in Laplace smoothing or normalize part, I made sure that the model does not overfit on tr1 and performs well enough on tr2. I tried alpha = 1 first. Then, alpha = 2, 0.1, 0.5, 0.6, 0.4, 0.7, 0.3.

The results showed that the model works best for alpha in range 0.3 to 0.7, and the best results were obtained at alpha =0.5 (accuracy almost 0.6).

Part 3: Train on entire dataset and test on Kaggle competition: Parameter tuning (more specific range)

Now, since I have a good approximation of alpha (range 0.3 to 0.7), I tried different values of alpha in this range. I took the entire training dataset to train our model, because that will give the best vocabulary size and probably the best results.

While running the code, the results are:

" Vocabulary Size: 65168

Training Set Size: 16000

Test Set Size: 4491

K is 5

alpha is 0.45

K is 65168

alpha is 0.45

trained

Evalution: Accuracy: 0.859250 Precision: 0.890121 Recall: 0.831020 Macro-F1: 0.856001 "

BY TRYING DIFFERENT VALUES OF ALPHA, I OBTAINED DIFFERENT RESULTS IN KAGGLE COMPETITION:

EXAMPLE:

Alpha, accuracy: (0.6, 0.59162), (0.5, 0.59697), (0.4, 0.59608), (0.45, 0.59875), (0.44, 0.59919)

The best result was obtained for alpha = 0.44 with score of 0.59919

Part 4: Submission of code with this report

I have only submitted the code for combined training dataset, and not of the split dataset. The only difference is that the code for training on split datasets has these more lines:

" grt_label = pd.read_csv("data/answer.csv")

acc, precision, recall, f1 = evaluate(grt_label["label"], test_data_pre)

print("Evalution: Accuracy: %f\tPrecision: %f\tRecall: %f\tMacro-F1: %f" % (acc, precision, recall, f1)) “

Since, I don’t have an answer file, thus I removed this code while using the entire training dataset (no split).

Part 5: Showing algorithms/run of code:

File is read and data is separated into parts (line 134) and read_data function is called, it tokenizes the text in words section (line 60) by calling the tokenize function, later a dictionary of words is made (lines 62-67). Now, data_matrix is created using get_bag_of_words function (line 69). Now, train_NB function is used to train our naïve bayes model (line 140), (train_NB uses the normalize function as well for Laplace smoothing, the normalize function has our alpha parameter). The working of naïve bayes model in this project is very similar to that taught to us in lab 4 and is thus self-explanatory. Now, train_data_pre (line 141) is used to make predictions on our training dataset, later evaluate function is used to evaluate the accuracy, precision, recall, Macro-F1 of our model on training dataset (used as test dataset).

Now, predictions are made on test dataset (using the above process till the training of NB, but we don’t train NB on test data, we have already trained NB on training data). Then we write our predictions in lab_submission.csv file.

In the split dataset case or when we have answer file as well, then we can evaluate our accuracy, precision, recall, Macro-F1 of our model on test dataset.

Later, based on the output results, we can tune our parameter alpha (lines 84,85,86).

Conclusion:

Lemmatization of tokens is useful, but not very much helpful in this project. Stopwords adjustment to have ‘no’ and ‘not’ included in the tokens was also not very much helpful. Still, they helped to some extent to improve the accuracy of our model.

In Naïve Bayes classifier, parameter tuning is very important and in my project I did it by splitting the training dataset into two parts of 13000 and 3000 entries and training my model on 13000 entries and testing on 3000 entries, and trying different alpha to find which alpha gives best accuracy. At the same time, I made sure that my model does not overfit on training data. Later, I used the entire training dataset of 16000 entries and applied parameter tuning (using the range of best alpha, obtained from analysis of parameter tuning on 13000 and 3000 entries).

Alpha, accuracy: (0.6, 0.59162), (0.5, 0.59697), (0.4, 0.59608), (0.45, 0.59875), (0.44, 0.59919)

The best result was obtained for alpha = 0.44 with score of 0.59919, which is above baseline 80.

THANKS FOR MARKING ☺