**Report**

**Yuhe CHANG**

**Basic information:**

Language: C++

Coding Platform: Visual Studio 2017

System: WIN10

**Architecture:**

For this task, I didn’t build my own classes. The whole program is process oriented. This program can read training.txt and test.txt, and use the data derived from training.txt to complete training process and labeling process.

The structure of my program is like this:

First, I read the training.txt and deposit all words into a *vector<map<string, int>>* variable. For this part, I implement BAG OF WORDS model. In other words, all Docs of one author become one Doc for me. So, there are 15 members in the *vector*, each *map<string, int>* contains the whole words that one author uses with their numbers of repetition.

Then I test the accuracy of training.txt. I choose some of words from each Doc as features. The so called “some of words” come from the modified Doc which filtered out words whose number of letters is fewer than the given THRESHOLD. I tried many different Threshold, after weighting the final accuracy and running time the program costs, I choose 4 as the THRESHOLD. The reason I didn’t let the program itself automatically pick the best THRESHOLD is because it will take so long to repeatedly test the accuracy of all 3000 training Docs. The principle of the prediction process is Naïve Bayes Classifier. I will further discuss it in **Model Building** part.

After that, I test the accuracy of test.txt. I use the same processes as I mention above.

The final part is the output part.

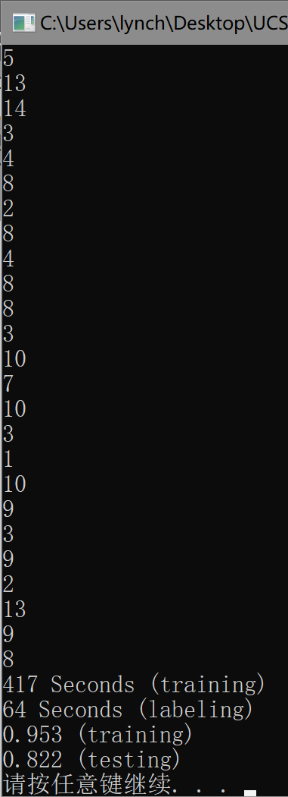
**Preprocessing:**

I filter out the words whose number of letters is fewer than the given THRESHOLD. I think this is like kind of stemming or something. So, the words having too few letters are considered meaningless for me.

**Model Building:**

In order to implement Naïve Bayes rule. I calculate the occurrence probability of each feature(word) in a Doc written by a certain author, the multiply them all up since we assume each feature is independent in NB Classifier. For the probability of occurrence of each author, I didn’t use the value of the number of Doc written by an author divided by the number of whole Docs. Instead, I choose the value of the number of words used by an author divided by the number of whole words. I didn’t calculate the denominator of the NB function since it is a constant for one prediction.

For the selection of THRESHOLD, I already explained it above. Actually, this is the training part in my program, but for the restriction of the time I didn’t let the program do the selection part.

**Results:**

I’m really sorry I didn’t run my program on the CSIL since I didn’t get the account as an extension student. Because of time confliction during midterm period, I didn’t find an appropriate time to do that. So, I only run it on my own PC whose CUP is i5. I attached the screenshot of my final result below.

FORMAT:

Predicted labels of testing.txt (n lines)

Training time

Labeling time

Training accuracy

Testing accuracy

**Challenges:**

1. The first challenge that I met is how to accelerate the whole process. At first, I used *vector* to store all data, but the program becomes too slow since *vector* only allow sequential search. Then, I change the container that I use. I use *map* afterwards since map uses red-black tree structure and the searching process become much faster.

2. The second challenge the I met is how to boost the accuracy. I fix it through comparing different THRESHOLD and changing the number of features that I choose.

About the 10 most important features for each class, I can’t list them out because I don’t know how to derive them since I choose all words from Doc that has been preprocessed (filtered words by THRESHOLD) as features for each test Doc.

**Weakness:**

1. the accuracy isn’t very high, which is around 82%. I think if applying other features, like the number of some sentence, the accuracy will be higher.

2. the running time is long, but I don’t know how to improve this situation.