|  |  |  |
| --- | --- | --- |
| **Discrete Mathematics** | Section | 02 |
| Student number | 21900156 |
| **HW6 – Spam filter** | Name | Kim, Ye Jun |

*If your explanation is less informative and insufficient, then you may not get any points.*

*Also, you should provide discussion, otherwise you will get penalty.*

* General informations

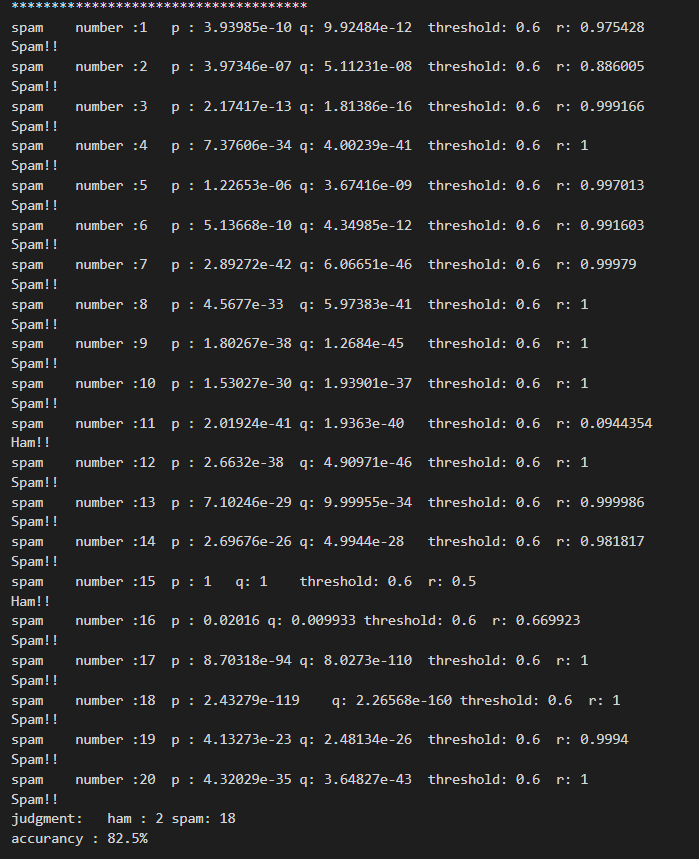
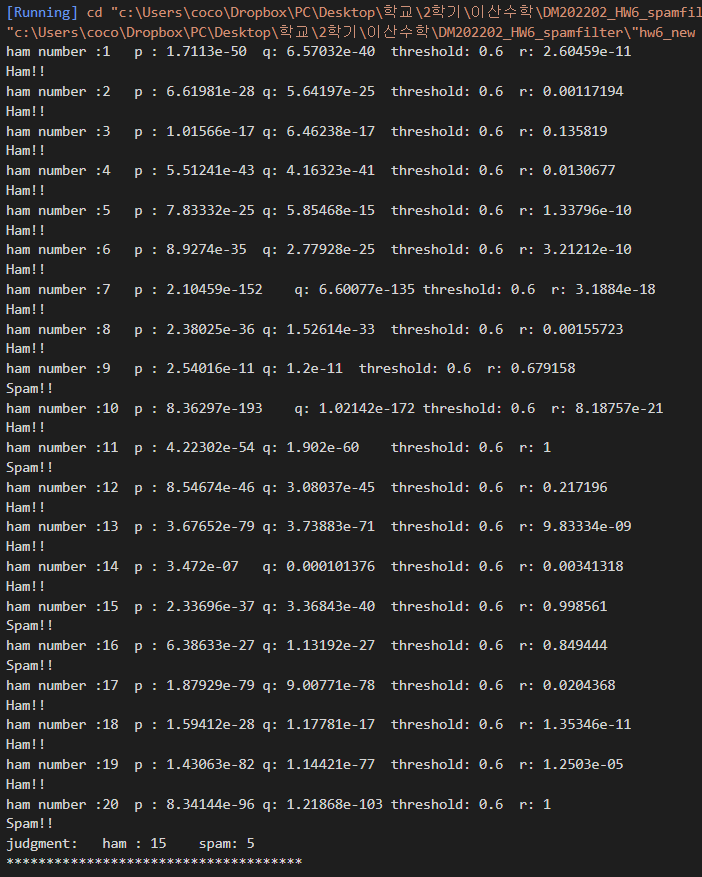
|  |  |
| --- | --- |
| Item | Your answer |
| The number of lines in your code. | 200 lines (include comments) |
| The number of functions in your code. | 5 functions(exclude main functions) |

* Functions

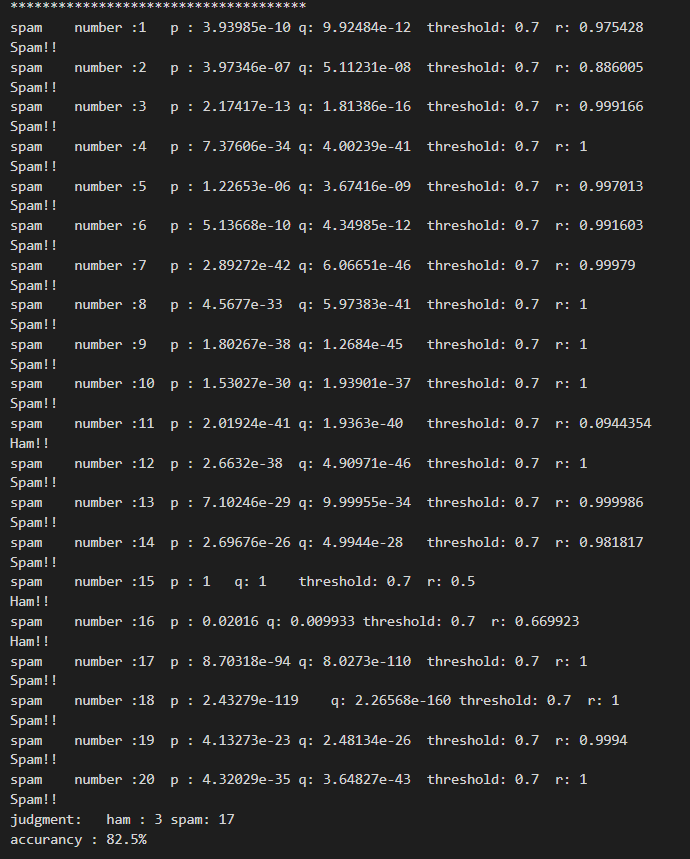
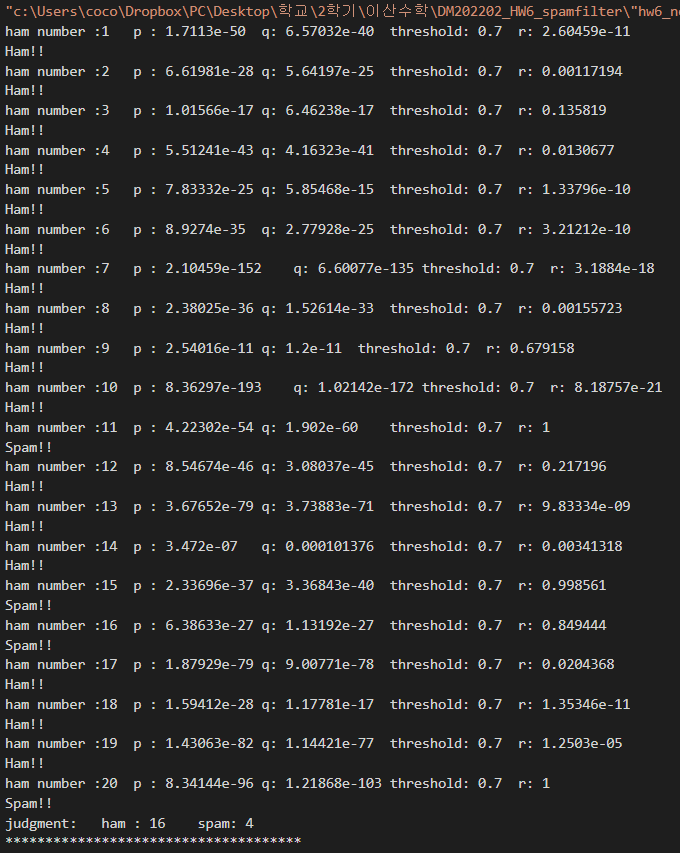
|  |  |
| --- | --- |
| Function name | Function Description |
| Void parse(string str, vector<string>& values,string& delimiter) | It is a function that classifies string “str” by the string “delimiter” as a delimiter, puts it in vector<string> “values”, and returns it. In the function, it finds the "delimiter" position in the "str", extracts the front part of the position, puts the extracted part in the "values", and erases the "delimiter" from the "str". In the last, just puts the ”str” in the “values” because there is no “delimiter”. |
| Vector<string> split(string text, char delimiter) | It is a function that classifies string “text” by the char “delimiter” as a delimiter and puts it in vector<string> “str\_vect” that declares in the function by using the push\_back() function. Then, return “str\_vect”. It is similar with the parse function, but it classifies string by character. |
| Void make\_wordnum(string input\_file, map<string, int> & count) | It is a function that organizes the words from the mail content in the file and stores how many times the words appear in the mail in map format “count”. In this function, it opens “input\_file” and only mail contents in “input\_file” are saved in “mail\_content” by using parse and substr, for functions. Then, it deletes non-word characters(eg,:,+=\..) and stores words in “split\_all\_words” by using split, erase, and remove functions. “split\_all\_word” stores a word just one time if it saves the same word in “split\_all\_word” because it is set. Then, it gets the mail in “mail\_conten” one by one and stores the words in the mail in “split\_word” in the same way as above so that they do not overlap. And if the word in the mail is the same as the word in the entire mail, increase it one by one and count the number of mails where the word came out. Then, “count” has the words as key and the number of mails where the word comes out as value. |
| Int test(string input\_file, map<string, int> count1, map<string,int> count2) | It is a function that calculates the probability calculated probability based on bayes theorem by using map “count1” and “count2” and determines whether the mails in “input\_file” are spam or not. In the same way as the “make\_wordnum” function, it gets the contents of the mail in the “input\_file” and classifies the words. Then, it takes the words one by one and checks if it is in the words of “count1” and “count2”, and multiplied by the probability if it is in the both of them. In this case, it calculates probability using bayes thoerom. Then, it judges which is spam by using “threshold”. If probability “r” is over the “threshold”, it is spam. Then, if it is “spam” type, it returns the number of “spam” which is judged as spam.Conversely, if it is “ham” type, it returns the number of “ham” which is judged as ham. |
| Double cal\_accuracy(int num\_ham, int num\_spam) | It is a function that receives the number of ham mail which is judged correctly and spam mail which is judged correctly as parameters, and calculates the probability of accuracy by using HAM\_TEST\_SIZE and SPAM\_TEST\_SIZE, which are defined in advance. |
| Int main(int argc, char\* argv[]) | It is a function that declares the file names to use, declares the ”word\_ham” variable that stores the words and the number of times from 100 ham mails by using the “make\_wordnum” function and declares the ”word\_spam” variable that stores the words and the number of times from 100 spam mails by using the “make\_wordnum” function. And it receives the number od correctly determined mails through the “test” function and uses the “cal\_accurancy” to calculate accurancy and prints it. |

* Screenshot of your program running

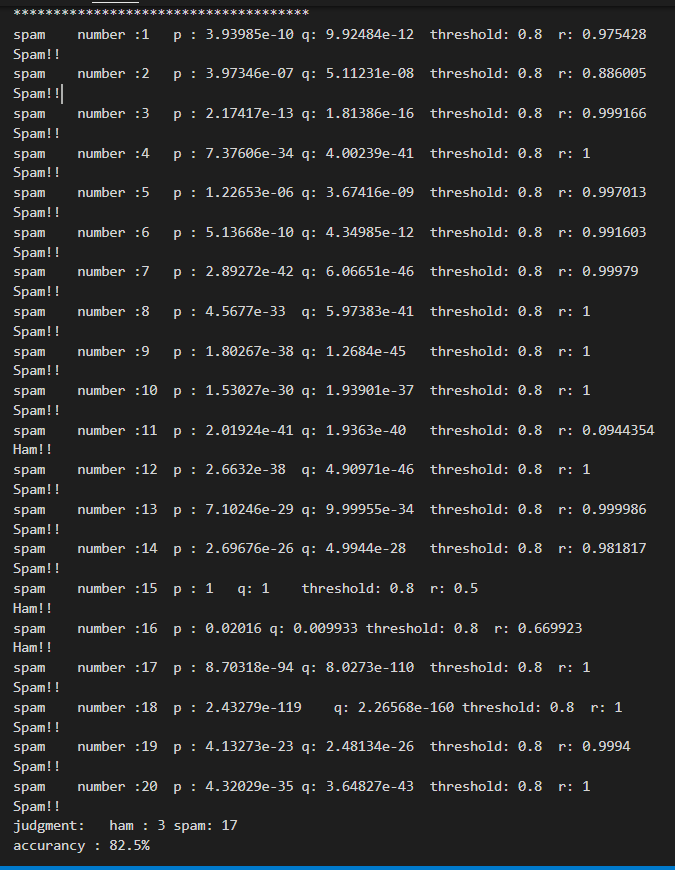
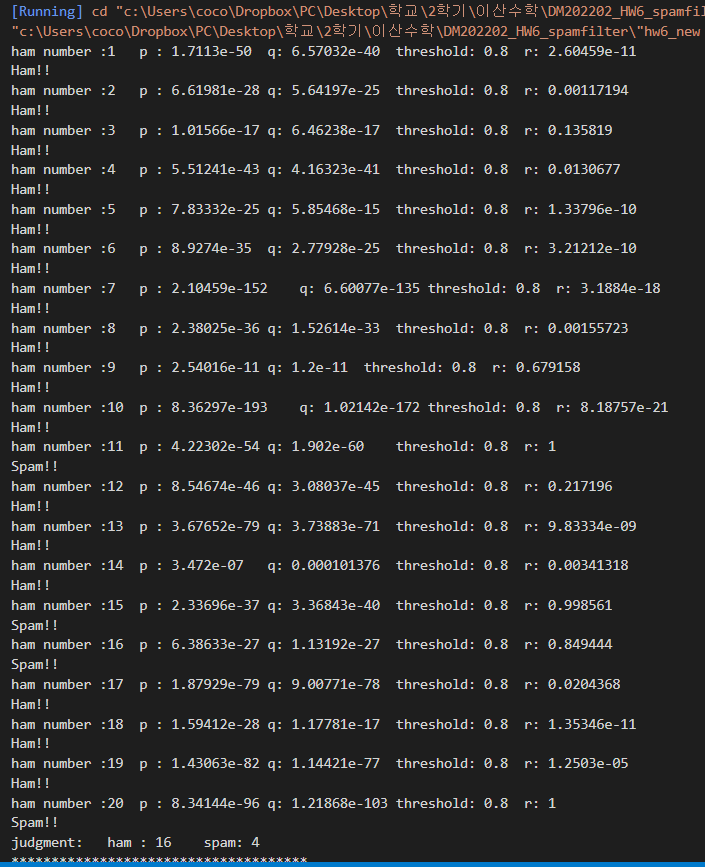
<threshold = 0.6>



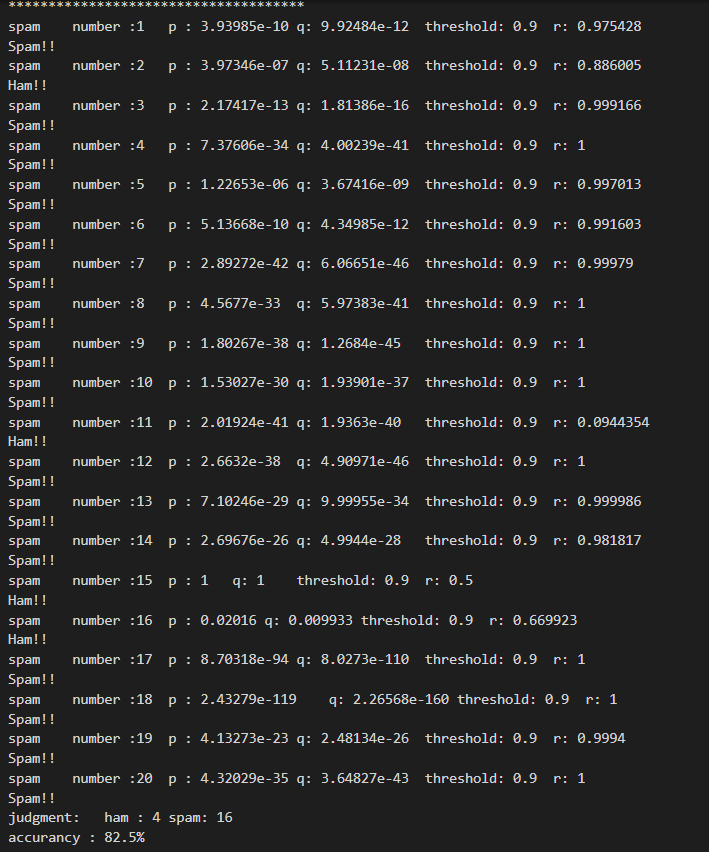
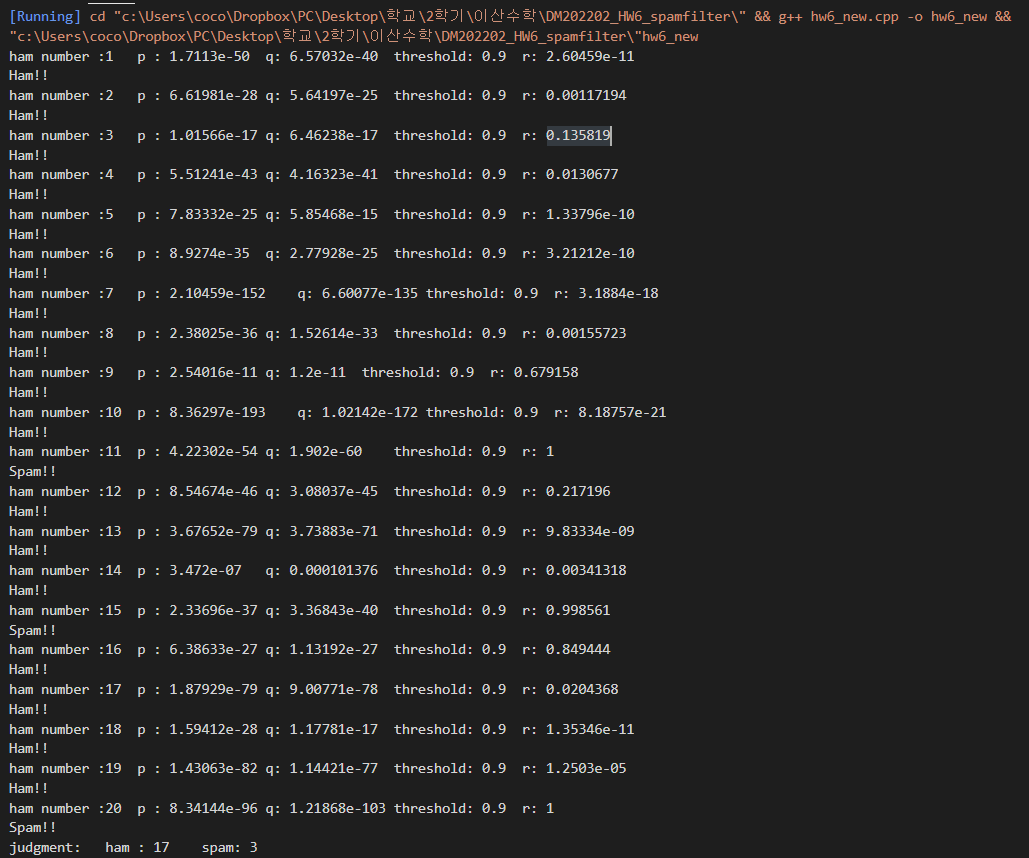
<threshold = 0.7>



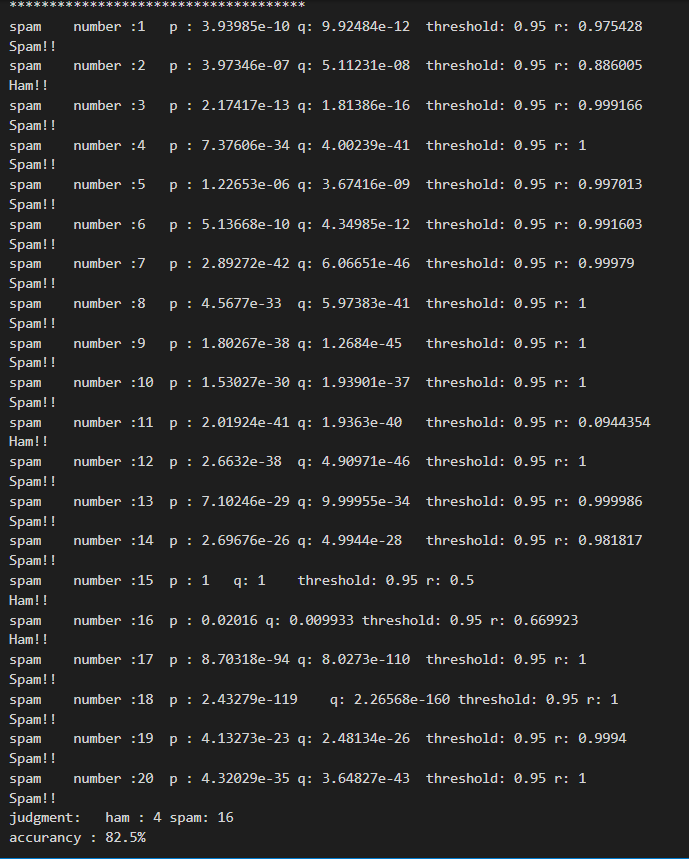
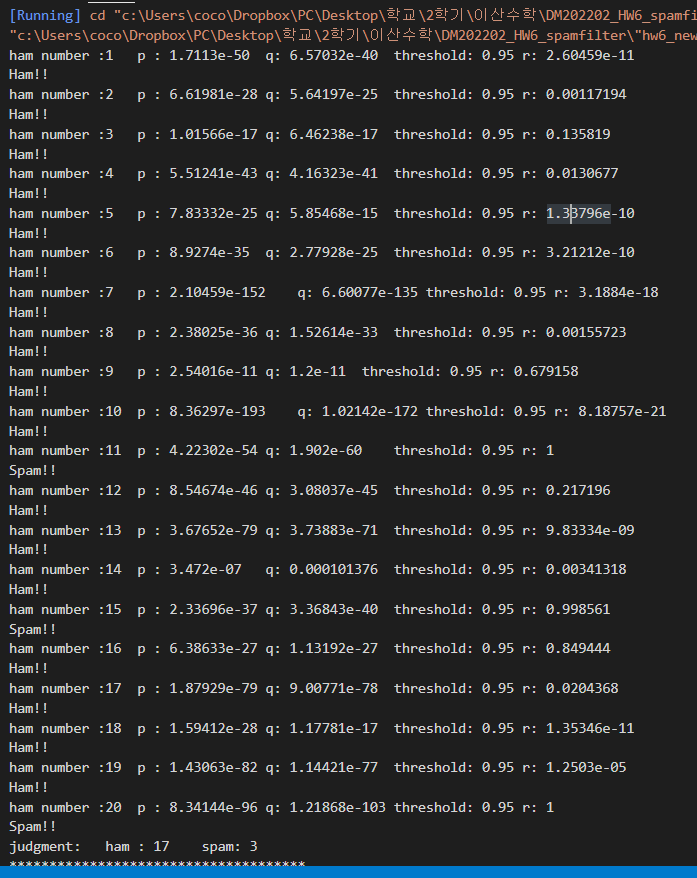
<threshold = 0.8>



<threshold = 0.9>



<threshold = 0.95>



|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| num  True | True Label | T = 0.6 | | T = 0.7 | | T = 0.8 | | T = 0.9 | | T = 0.95 | |
|  | predicted  label |  | predicted  label |  | predicted  label |  | predicted  label |  | predicted  label |
| s01 | Spam | 0.975428 | spam | 0.975428 | spam | 0.975428 | spam | 0.975428 | spam | 0.975428 | spam |
| s02 | Spam | 0.886005 | spam | 0.886005 | spam | 0.886005 | Spam | 0.886005 | Ham | 0.886005 | Ham |
| s03 | Spam | 0.999166 | spam | 0.999166 | spam | 0.999166 | Spam | 0.999166 | spam | 0.999166 | spam |
| s04 | Spam | 1 | spam | 1 | spam | 1 | Spam | 1 | spam | 1 | spam |
| s05 | Spam | 0.997013 | Spam | 0.997013 | Spam | 0.997013 | Spam | 0.997013 | Spam | 0.997013 | Spam |
| s06 | Spam | 0.991603 | Spam | 0.991603 | Spam | 0.991603 | Spam | 0.991603 | Spam | 0.991603 | Spam |
| s07 | Spam | 0.99979 | Spam | 0.99979 | Spam | 0.99979 | Spam | 0.99979 | Spam | 0.99979 | Spam |
| s08 | Spam | 1 | Spam | 1 | Spam | 1 | Spam | 1 | Spam | 1 | Spam |
| s09 | Spam | 1 | Spam | 1 | Spam | 1 | Spam | 1 | Spam | 1 | Spam |
| s10 | Spam | 1 | Spam | 1 | Spam | 1 | Spam | 1 | Spam | 1 | Spam |
| s11 | Spam | 0.0944354 | Ham | 0.0944354 | Ham | 0.0944354 | Ham | 0.0944354 | Ham | 0.0944354 | Ham |
| s12 | Spam | 1 | Spam | 1 | Spam | 1 | Spam | 1 | Spam | 1 | Spam |
| s13 | Spam | 0.999986 | Spam | 0.999986 | Spam | 0.999986 | Spam | 0.999986 | Spam | 0.999986 | Spam |
| s14 | Spam | 0.981817 | Spam | 0.981817 | Spam | 0.981817 | Spam | 0.981817 | Spam | 0.981817 | Spam |
| s15 | Spam | 0.5 | Ham | 0.5 | Ham | 0.5 | Ham | 0.5 | Ham | 0.5 | Ham |
| s16 | Spam | 0.669923 | spam | 0.669923 | Ham | 0.669923 | Ham | 0.669923 | Ham | 0.669923 | Ham |
| s17 | Spam | 1 | Spam | 1 | Spam | 1 | Spam | 1 | Spam | 1 | Spam |
| s18 | Spam | 1 | Spam | 1 | Spam | 1 | Spam | 1 | Spam | 1 | Spam |
| s19 | Spam | 0.9994 | Spam | 0.9994 | Spam | 0.9994 | Spam | 0.9994 | Spam | 0.9994 | Spam |
| s20 | Spam | 1 | spam | 1 | spam | 1 | Spam | 1 | spam | 1 | spam |
| h01 | Non-spam | 2.60459e-11 | Ham | 2.60459e-11 | Ham | 2.60459e-11 | Ham | 2.60459e-11 | Ham | 2.60459e-11 | Ham |
| h02 | Non-spam | 0.00117194 | Ham | 0.00117194 | Ham | 0.00117194 | Ham | 0.00117194 | Ham | 0.00117194 | Ham |
| h03 | Non-spam | 0.135819 | Ham | 0.135819 | Ham | 0.135819 | Ham | 0.135819 | Ham | 0.135819 | Ham |
| h04 | Non-spam | 0.0130677 | Ham | 0.0130677 | Ham | 0.0130677 | Ham | 0.0130677 | Ham | 0.0130677 | Ham |
| h05 | Non-spam | 1.33796e-10 | Ham | 1.33796e-10 | Ham | 1.33796e-10 | Ham | 1.33796e-10 | Ham | 1.33796e-10 | Ham |
| h06 | Non-spam | 3.21212e-10 | Ham | 3.21212e-10 | Ham | 3.21212e-10 | Ham | 3.21212e-10 | Ham | 3.21212e-10 | Ham |
| h07 | Non-spam | 3.1884e-10 | Ham | 3.1884e-10 | Ham | 3.1884e-10 | Ham | 3.1884e-10 | Ham | 3.1884e-10 | Ham |
| h08 | Non-spam | 0.00155723 | Ham | 0.00155723 | Ham | 0.00155723 | Ham | 0.00155723 | Ham | 0.00155723 | Ham |
| h09 | Non-spam | 0.679158 | spam | 0.679158 | Ham | 0.679158 | Ham | 0.679158 | Ham | 0.679158 | Ham |
| h10 | Non-spam | 8.18757e-21 | Ham | 8.18757e-21 | Ham | 8.18757e-21 | Ham | 8.18757e-21 | Ham | 8.18757e-21 | Ham |
| h11 | Non-spam | 1 | Spam | 1 | Spam | 1 | Spam | 1 | Spam | 1 | Spam |
| h12 | Non-spam | 0.217196 | Ham | 0.217196 | Ham | 0.217196 | Ham | 0.217196 | Ham | 0.217196 | Ham |
| h13 | Non-spam | 9.83334e-09 | Ham | 9.83334e-09 | Ham | 9.83334e-09 | Ham | 9.83334e-09 | Ham | 9.83334e-09 | Ham |
| h14 | Non-spam | 0.00341318 | Ham | 0.00341318 | Ham | 0.00341318 | Ham | 0.00341318 | Ham | 0.00341318 | Ham |
| h15 | Non-spam | 0.998561 | Spam | 0.998561 | Spam | 0.998561 | Spam | 0.998561 | Spam | 0.998561 | Spam |
| h16 | Non-spam | 0.849444 | spam | 0.849444 | spam | 0.849444 | spam | 0.849444 | Ham | 0.849444 | Ham |
| h17 | Non-spam | 0.0204368 | Ham | 0.0204368 | Ham | 0.0204368 | Ham | 0.0204368 | Ham | 0.0204368 | Ham |
| h18 | Non-spam | 1.35346e-11 | Ham | 1.35346e-11 | Ham | 1.35346e-11 | Ham | 1.35346e-11 | Ham | 1.35346e-11 | Ham |
| h19 | Non-spam | 1.2503e-05 | Ham | 1.2503e-05 | Ham | 1.2503e-05 | Ham | 1.2503e-05 | Ham | 1.2503e-05 | Ham |
| h20 | Non-spam | 1 | Spam | 1 | Spam | 1 | Spam | 1 | Spam | 1 | Spam |
| Accuracy (%) | | 82.5 | | 82.5 | | 82.5 | | 82.5 | | 82.5 | |

* Results
  + Probability and predicted label (spam or non-spam) for different threshold (T) (0.6, 0.7, 0.8, 0.9 and 0.95)

※ Accuracy is calculated from the following equation.

* Discussion (your interpretation of the results and possible strategy to improve the algorithm)

Even if I changed the threshold, the accuracy is not changed. The accuracy of spam is changed and the accuracy of ham is changed, but the total accuracy is not changed. I think that by changing the threshold, it can change the judgment of spam mail that is not judged correctly, but it also can change the judgment of ham mail that is judged correctly. So, I guess that it doesn’t change the accuracy.

I just divided the words in the mail into spaces in the code, but I think it would be good if the code doesn’t consider words such as “t”, “s”, and “xs” as words. And I thought that if code combines words like “play”, “played”, and “playing” into one word “play”, it can organize the words faster. And I think that extracting only mail’s content from csv file took a long time using various functions, so it would be better to etract this part faster.

□ Codes

// Put code here, and you should also submit your original executable C code.

// If C code does not run, then No point will be given.

#include <iostream>

#include <fstream>

#include <map>

#include <set>

#include <sstream>

#include <string>

#include <vector>

#include <regex>

using namespace std;

#define HAM\_TRAIN\_SIZE 100

#define SPAM\_TRAIN\_SIZE 100

#define HAM\_TEST\_SIZE 20

#define SPAM\_TEST\_SIZE 20

//string delimiter를 구분자로 해서 구분해 vector<string>values에 넣어주는 함수이다.

void parse(string str, vector<string>& values, string& delimiter){

    int pos =0;

    string token;

    while((pos= str.find(delimiter))!=string::npos){

        token = str.substr(0,pos);

        values.push\_back(token);

        str.erase(0,pos+delimiter.length());

    }

    values.push\_back(str);

}

//char를 구분자로 해서 vector로 반환해주는 함수이다.

vector<string> split(string text, char delimiter)

{

    {

        vector<string> str\_vect;

        stringstream input(text);

        string temp;

        while (getline(input, temp, delimiter))

        {

            str\_vect.push\_back(temp);

        }

        return str\_vect; //delimiter로 나눈 string을 저장하는 vector를 반환해준다.

    }

}

//input\_file에 있는 단어와 단어가 들어가 있는 메일 수를 map type에 저장해주는 함수이다.

void make\_wordnum(string input\_file, map<string, int>& count){

    string punctuation="+\'=\\`~!@#$%^&\*()-<>?/.,:;-\_{}[]|\n\"";//지어줘야하는 문자들이다.

    vector<string> mail\_content; // 메일내용만을 담고 있는 vector이다.

    vector<string> mail\_prev; //," 로 나눈 문자이다.

    vector<string> just\_word; //" "를 단위로 나누어진 단어를 임시로 저장한다. 가공되기 전 단어를 받는다.

    set<string> split\_all\_word; //필요없는 문자들을 제거하고 unique한 단어들을 저장한다.

    set<string> split\_word; //문장단위로 있는 단어를 저장한다.

    string wholeline;//모든 문서를 한번에 받는 string이다.

    ifstream file(input\_file);

    stringstream ss;

    ss << file.rdbuf();

    wholeline = ss.str();//파일의 전체내용을 string형식으로 저장해준다.

    string delimiter(",\"");

    parse(wholeline,mail\_prev,delimiter); // ,"를 기준으로 구분해준다.

    mail\_prev.erase(mail\_prev.begin()); //처음에 숫자와 메일유형이 오기 때문에 이를 지워준다.

    for(string value : mail\_prev){

        value=value.substr(0,value.rfind("\"")); //뒤에서부터 "를 찾아 그 뒤에 있는 메일번호와 메일유형은 필요하지 않기 때문에 지워준다.

        mail\_content.push\_back(value); //이제 메일내용만을 담고 있기 때문에 이를 mail\_content에 담아준다.

        if(value.find("\n")!=string::npos)

            value=regex\_replace(value,regex("\n")," "); //개행문자는 구분을 못해주어서 개행문자를 " "(공백)으로 바꾸어주었다.

        just\_word = split(value, ' '); // ' '(공백)을 기준으로 가공하기 전의 단어들로 저장해준다.

         for(string w : just\_word){

            for (char c: punctuation) {

                w.erase(remove(w.begin(), w.end(), c), w.end()); //필요없는 문자를 제거해준다.

            }

            if(w!="")

            split\_all\_word.insert(w);

        } // 가공하기 전의 단어들을 가져와 필요없는 문자들을 제거하고 set에 넣어주어 중복없는 단어들로 정리한다.

    }

    for(string line : mail\_content){

        if(line.find("\n")!=string::npos)

            line=regex\_replace(line,regex("\n")," "); //개행문자는 구분을 못해주어서 개행문자를 " "(공백)으로 바꾸어주었다.

        just\_word = split(line, ' ');

        for(string w : just\_word){

            for (char c: punctuation) {

                    w.erase(remove(w.begin(), w.end(), c), w.end()); //필요없는 문자를 제거해준다.

            }

            if(w!=""){

                split\_word.insert(w);

            }

        } //메일내용에서 필요없는 문자들을 제거하고 중복없는 단어로 정리해 set에 넣어준다. (split\_word는 하나의 메일에서 나온 단어들이다.)

        for(string word:split\_word){

           for(string word\_all:split\_all\_word){

                if(word==word\_all)

                count[word\_all]= count[word\_all]+1;// 메일에 총 메일에서 나온 단어가 존재하면 개수를 하나씩 올려 map 변수에 저장해준다.

           }

        }

        split\_word.clear();//메일마다 달라지기 때문에 비워주어야한다.

    }//하나의 메일 단위로 단어를 받아와 총 메일의 그 단어가 몇 번 있는지를 저장해준다.

}

//test를 해주는 함수로 확률을 계산해서 spam 메일인지 아닌지를 판단해주는 함수이다.

int test(string input\_file, map<string, int> count1, map<string, int > count2){

    string punctuation="+\'=\\`~!@#$%^&\*()-<>?/.,:;-\_{}[]|\n\"";//지어줘야하는 문자들이다.

    vector<string> mail\_prev; //," 로 나눈 문자이다.

    vector<string> just\_word; //" "를 단위로 나누어진 단어를 임시로 저장한다. 가공되기 전 단어를 받는다.

    set<string> split\_word; //문장단위로 있는 단어를 저장한다.

    string wholeline;//모든 문서를 한번에 받는 string이다.

    double threshold =0.6; //허용해주는 범위를 저장하는 변수이다.

    int num= 1; //번호를 표현하기 위해 사용한다.

    int count\_ham=0; //non-spam메일로 판단한 개수를 세기위해 선언한 변수이다.

    int count\_spam= 0; //spam 메일로 판단한 개수를 세기위해 선언한 변수이다.

    bool spam\_check=false;//유형이 spam인지 확인한다.

    bool ham\_check=false; //유형이 ham인지 확인한다.

    string type;//어느 유형인지를 저장하는 변수이다.

    ifstream file(input\_file);

    stringstream ss;

    ss << file.rdbuf();

    wholeline = ss.str();

    string delimiter(",\"");

    parse(wholeline,mail\_prev,delimiter); // ,"를 기준으로 구분해준다.

    if(mail\_prev[0].find("spam")!=string::npos)

        spam\_check= true;

    else

        ham\_check= true;

    mail\_prev.erase(mail\_prev.begin()); //처음에 숫자와 메일유형이 오기 때문에 이를 지워준다.

    for(string value : mail\_prev){

        value=value.substr(0,value.rfind("\"")); //뒤에서부터 "를 찾아 메일번호와 메일유형을 필요하지 않기 때문에 지워준다.

        if(value.find("\n")!=string::npos)

            value=regex\_replace(value,regex("\n")," "); //개행문자는 구분을 못해주어서 개행문자를 " "(공백)으로 바꾸어주었다.

        just\_word = split(value, ' ');

         for(string w : just\_word){

            for (char c: punctuation) {

                w.erase(remove(w.begin(), w.end(), c), w.end()); //필요없는 문자를 제거해준다.

            }

            if(w!="")

            split\_word.insert(w);

        } //필요없는 문자들을 제거하고 중복없는 단어로 정리해 set에 넣어준다.

        double p =1.0;

        double q =1.0;

        double r =0.0;

        for(string word : split\_word){

            if((count1[word]!=0) && (count2[word]!=0)){

                p \*=((double) count2[word]/ SPAM\_TRAIN\_SIZE);

                q \*=((double) count1[word]/ HAM\_TRAIN\_SIZE);

            }

        }

        r= p / (p+q);

        if(ham\_check) type= "ham";

        else type= "spam";

        cout <<type << "\tnumber :"<<num  << "\tp : "<< p <<"\tq: "<< q <<"\tthreshold: "<<threshold << "\tr: "<< r <<endl;

        if (threshold > r) {

            cout << "Ham!!" << endl;

            count\_ham++;

        }

        else {

            cout << "Spam!!" << endl;

            count\_spam++;

        }

        split\_word.clear();//비워준다.

        num++;

    }

    cout<<"judgment:\t"<<"ham : "<<count\_ham<< "\tspam: "<<count\_spam<<endl;

    if(type == "ham")

        return count\_ham;

    else

        return count\_spam;

}

//정확도를 계산해주는 함수이다.

double cal\_accuracy(int num\_ham, int num\_spam){

    return ((double)(num\_ham+num\_spam))/(HAM\_TEST\_SIZE+SPAM\_TEST\_SIZE)\*100;

}

int main(int argc, char\* argv[]){

    map<string, int> word\_ham; //ham에 있는 단어와 횟수를 저장해준다.

    map<string, int> word\_spam; //spam에 있는 단어와 횟수를 저장한다.

    string ham\_file = ".\\csv\\train\\dataset\_ham\_train100.csv";

    string spam\_file = ".\\csv\\train\\dataset\_spam\_train100.csv";

    string test\_ham\_file = ".\\csv\\test\\dataset\_ham\_test20.csv";

    string test\_spam\_file = ".\\csv\\test\\dataset\_spam\_test20.csv";

    make\_wordnum(ham\_file,word\_ham); //non-spam mail의 단어와 나온 개수 저장

    make\_wordnum(spam\_file, word\_spam); //spam mail의 단어와 나온 개수 저장

    int ham = test(test\_ham\_file, word\_ham, word\_spam);

    cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

    int spam = test(test\_spam\_file, word\_ham, word\_spam);

    double accuracy = cal\_accuracy(ham, spam);

    cout<<"accurancy : "<<accuracy<<"%"<<endl;

}