ASSIGNMENT 4
Naïve Bayes

Implement a naïve bayesian learning method to detect if a document is spam or not.

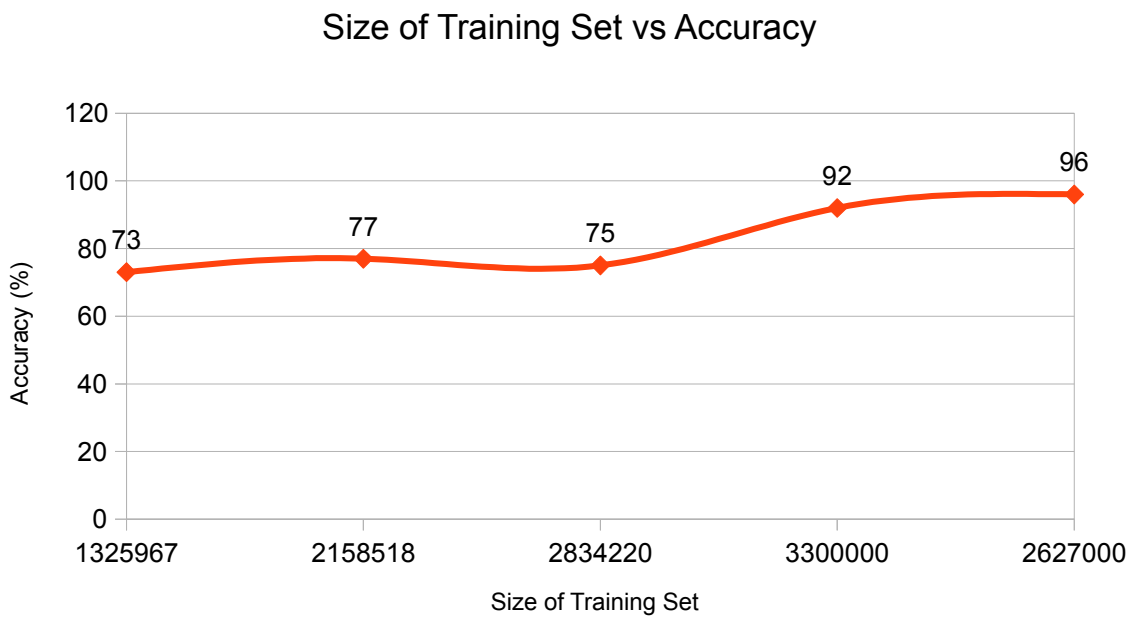
I have implemented a set to feed the data set into a map. I then take the input data and compute the log probabilities since the normal probabilities will be very less.

THE ALGORITHM

My code first asks for the user to train the computer on Spam and Non-Spam data. Then, I compute the probability of the training data.

GRAPH

I have shown the graph of accuracy versus the size of the training set



Source Code

```
/*
 Author: Himanshu Jindal
 Description: AI Assignment 4
 The program asks for user to enter data for spam and not spam mails.
 It then decides whether a given mail entered is spam or not spam based on the training it has received
 */
#include <fstream>
```

```
#include <iostream>
#include <string>

#include <dirent.h>
#include <unistd.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <map>
#include <math.h>

//#define DEBUG
//#define DEBUG1
using namespace std;

struct node
{
    int spam;
    int not_spam;
};

const char _DELIM_ = ' ';
const char _NEW_LINE_ = '\n';
const char _TAB_ = '\t';

map<string, node> Prob_Table;
int SPAM_WORDS=0;
int NON_SPAM_WORDS=0;

void TakeInput();
void Read_Dir(string, bool);
void addData(string, bool);
void addWord(string, bool);
bool is_delim(char);
int Compute_Prob(string);
int GetWord(string, bool);

int main()
{
    int ch=1;
    bool repeat = true;
    while(repeat==true)
    {
        TakeInput();
        cout<<"\nDo you want to enter more data?\n1.Yes\n2.No\nEnter Choice(1 or 2):";
        cin>>ch;
        if(ch==1)
```

```
            repeat = true;
        else
            repeat = false;
    }
    string file_name;
    cout<<"\nEnter the name of the file whose probabilty of spam/not spam is to be calculated:";
    cin>>file_name;
    double result = Compute_Prob(file_name);
    if(result==1)
        cout<<"\nSPAM!!!";
    else
        if(result==0)
            cout<<"\nNOT SPAM!!";
    else
        cout<<"CLASH.. CANT SAY";
    return 0;
}

int Compute_Prob(string dir_name)
{
    double Prob_Spam = log(((double)(::SPAM_WORDS))/((double)((::SPAM_WORDS)+
(::NON_SPAM_WORDS))));
    double Prob_NotSpam = log(((double)(::NON_SPAM_WORDS))/((double)((::SPAM_WORDS)+
(::NON_SPAM_WORDS))));
    ifstream myFile(dir_name.c_str());
    if(myFile.is_open())
    {
        while(myFile.good())
        {
            string curLine;
            getline(myFile, curLine);
#ifdef DEBUG
            cout<<"\nRead line:"<<curLine;
#endif
            bool in_delim = false;
            string str;
            int i = 0;
            do
            {
                bool is_delimiter = is_delim(curLine.c_str()[i]);
                if(in_delim == false && is_delimiter==false)
                {
                    str.append(1, curLine.c_str()[i]);
                }
                else if(in_delim == false && is_delimiter == true)
                {
```

```
                if(!str.empty())
                {
#ifdef DEBUG1
                    cout<<"\nComputing prob for word:"<<str;
#endif
                    int occ_spam = GetWord(str, true);
                    int occ_not_spam = GetWord(str, false);
                    double temp_prob_spam, temp_prob_not_spam;
                    if(occ_spam!=0)
                    {
                        temp_prob_spam = (((double)occ_spam))/((double)(::SPAM_WORDS));
                    }
                    else
                    {
                        temp_prob_spam = (((double)occ_spam)+1)/((double)(2*(::SPAM_WORDS)+
(::NON_SPAM_WORDS)));

                    }
                    if(occ_not_spam!=0)
                    {
                        temp_prob_not_spam = (((double)occ_not_spam))/((double)
(::NON_SPAM_WORDS));
                    }
                    else
                    {
                        temp_prob_not_spam = (((double)occ_not_spam)+1)/((double)
(2*(::NON_SPAM_WORDS)+(::SPAM_WORDS)));

                    }
                    Prob_Spam += log(temp_prob_spam);
                    Prob_NotSpam += log(temp_prob_not_spam);
#ifdef DEBUG1

                    cout<<"\nOccurence of word in spam->"<<occ_spam
                    <<"\nProb_Spam total till now->"<<(Prob_Spam)
                    <<"\nOccurence of word in non spam->"<<occ_not_spam
                    <<"\nProb_notSpam till now->"<<(Prob_NotSpam);
#endif
                    str.clear();
                }
                else
                {
                }
                in_delim = true;
            }
            else if(in_delim==true && is_delimiter == false)
```

```
            {
                str.append(1, curLine.c_str()[i]);
                in_delim = false;
            }
            else if(in_delim == true && is_delimiter == true)
            {

            }
        }while(curLine[i++]!='\0');

    }
  }
  cout<<"\nLog Probabilty of being spam is:"<<Prob_Spam;
  cout<<"\nLog Probabilty of not being spam is:"<<Prob_NotSpam;
  if(Prob_Spam>Prob_NotSpam)
    return 1;
  else if(Prob_Spam<Prob_NotSpam)
  {
    return 0;
  }
  else
    return -1;
}

void TakeInput()
{
  bool repeat = true;
  int ch=0;
  string path;
  do
  {
    cout<<"\nLet us train our computer."
    <<"\nWhat kind of data do you want to enter"
    <<"\n1. Spam\n2. Non-Spam\nEnter choice(1,2):";
    cin>>ch;
    if(ch!=1 && ch!=2)
    {
      repeat = true;
      cout<<"\n!!! Please Enter correct value";
    }
    else
      repeat = false;
  } while (repeat == true);
  cout<<"\nEnter the path where this data is stored:";
  cin>>path;
  bool Spam = (ch==1)?true:false;
```

```
    Read_Dir(path, Spam);
}

void Read_Dir(string path, bool Spam)
{
    string retString;
    ifstream fin;
    string filepath;
    int num;
    DIR *dp;
    struct dirent *dirp;
    struct stat filestat;

    dp = opendir( path.c_str() );
    if (dp == NULL)
    {
        cout << "Error() opening " << path << endl;
        return;
    }
    else
    {
#ifdef DEBUG
        cout<<"\nDirectory opened";
#endif
    }
    while ((dirp = readdir( dp )))
    {
        filepath = path + "/" + dirp->d_name;
#ifdef DEBUF
        cout<<"\nOpening File:"<<filepath;
#endif
        // If the file is a directory (or is in some way invalid) we'll skip it
        if (stat( filepath.c_str(), &filestat )) continue;
        if (S_ISDIR( filestat.st_mode ))        continue;

        // Endeavor to read a single number from the file and display it
        fin.open( filepath.c_str() );
        while(fin.good())
        {
            string curLine;
            getline(fin, curLine);
#ifdef DEBUG
            cout<<"\nRead line:"<<curLine;
#endif
            addData(curLine, Spam);
        }
```

```
        /*if (fin >> num)
         cout << filepath << ": " << num << endl;
         */
        fin.close();
    }
#ifdef DEBUG
    cout<<"\nAll Files read. Closing directory";
#endif
    closedir( dp );
}

void addData(string curLine, bool Spam)
{
    int i = 0;
    bool in_delim = false;
    string str;
    do
    {
        bool is_delimiter = is_delim(curLine.c_str()[i]);
        if(in_delim == false && is_delimiter==false)
        {
            str.append(1, curLine.c_str()[i]);
        }
        else if(in_delim == false && is_delimiter == true)
        {
            if(!str.empty())
            {
#ifdef DEBUG
                cout<<"\nAdding word:"<<str;
#endif
                addWord(str, Spam);
                str.clear();
            }
            else
            {
            }
            in_delim = true;
        }
        else if(in_delim==true && is_delimiter == false)
        {
            str.append(1, curLine.c_str()[i]);
            in_delim = false;
        }
        else if(in_delim == true && is_delimiter == true)
        {
```

```
        }
    }while(curLine[i++]!='\0');
}

void addWord(string str, bool Spam)
{
    if(Prob_Table.count(str)>0)
    {
#ifdef DEBUG
        cout<<"\n"<<str<<" already exists in map";
#endif
        node temp = Prob_Table[str];
        map<string, node>::iterator it;
        it = Prob_Table.find(str);
        Prob_Table.erase(it);
        if(Spam==true)
        {
            temp.spam++;
            (::SPAM_WORDS)++;
#ifdef DEBUG
            cout<<"\nSpam->"<<temp.spam<<"\tTotal Spam:"<<::SPAM_WORDS;
#endif
        }
        else
        {
            temp.not_spam++;
            (::NON_SPAM_WORDS)++;
#ifdef DEBUG
            cout<<"\nNot Spam->"<<temp.not_spam<<"\tTotal Not Spam:"<<::NON_SPAM_WORDS;
#endif
        }
        Prob_Table.insert(pair<string,node>(str, temp));
    }
    else
    {
        node temp = {0,0};
        if(Spam==true)
        {
            temp.spam++;
            (::SPAM_WORDS)++;
#ifdef DEBUG
            cout<<"\nSpam->"<<temp.spam<<"\tTotal Spam:"<<::SPAM_WORDS;
#endif
        }
        else
        {
```

```
            temp.not_spam++;
            (::NON_SPAM_WORDS)++;
#ifdef DEBUG
            cout<<"\nNot Spam->"<<temp.not_spam<<"\tTotal Not Spam:"<<::NON_SPAM_WORDS;
#endif
        }
        Prob_Table.insert(pair<string,node>(str, temp));
    }
}

bool is_delim(char c)
{
    if(c==' '|| c=='\t' || c=='\n'|| c=='\0')
        return true;
    else return false;
}

int GetWord(string str, bool spam)
{
    if(Prob_Table.count(str)>0)
    {
        node temp = Prob_Table[str];
        if(spam==true)
            return temp.spam;
        else
            return temp.not_spam;
    }
    else
        return 0;
}
```