



INDIAN INSTITUTE OF TECHNOLOGY, GUWAHATI

Department of Computer Science and Engineering

Project Report on

SPEECH THERAPY

Based on Speech Recognition System

Submitted to:

Prof. P. K. Das

Submitted by:

Adwaith R Krishna (224101005)

Kartikay Bhardwaj (224101030)

Neha Amrit Dhuttargaon (224101037)

For course fulfilment of CS566: Speech Processing

ACKNOWLEDGEMENT

This project is being submitted as a requirement for course fulfilment of CS566 - Speech Processing. It is a pleasure to acknowledge our sense of gratitude to Prof. P.K. Das who guided us throughout the project work. His timely guidance and suggestions were encouraging. We thank to the Teaching Assistants who were always helpful in clearing doubts. Finally, we thank to our classmates for the support.

1. Adwaith R Krishna (224101005)
2. Kartikay Bhardwaj (224101030)
3. Neha Amrit Dhuttargaon (224101037)

Contents

1	Abstract	4
2	Introduction	4
2.1	What is Speech Recognition	4
2.2	Our Project	4
2.3	Future improvements	4
3	Experimental Setup	4
4	Proposed Techniques	5
4.1	Flowchart	5
4.2	Model description	5
5	Result	6
5.1	Home Page	6
5.2	Live Scoring	6
5.3	Live Training	6
6	Source Code	12
6.1	test.h	12
6.2	Form1.h	62

List of Figures

1	Flowchart of the project	5
2	Home Page	7
3	Live Scoring Section	8
4	Live Training	9
5	Live Training with new words	10
6	Recording Console	11

1 Abstract

This project is developed using C++/C. It can take a speech sample for 2 seconds, preferably the words displayed on the screen and display its corresponding score matching against a well spoken therapist voice. Initially it is developed for simple words which can be used as a tutorial for kids. But it can be expanded further for a bigger area of words. It uses the concepts of the famous Hidden Markov Model to store the properties of the speech sample and compare the new sample with these properties to detect which word has been spoken.

2 Introduction

2.1 What is Speech Recognition

Speech Recognition is a technique which is quite popular now-a-days. When we speak into a microphone which is connected to the computer/mobile, it converts it to a text file which contains some amplitude values. Those values are basically the deviation of the speech signal from X-axis. Then we can use this file, do some calculations which can detect which word has been spoken and then further steps can be taken as per the requirement. One such application is Alexa.

2.2 Our Project

Our project is based on the concept of scoring against the well-trained voice of a therapist to help children learn the correct pronunciation of certain words. Here we have limited ourselves to three words Apple, Tomato, and Orange. We can improve it to include multiple words in the future. We are storing the scores for tracking the improvement.

2.3 Future improvements

In the future in order to extend the usability of the app, we can add a live training feature where the therapist can add new words without a help of a developer. This is a desktop application making it a web application will improve its accessibility.

3 Experimental Setup

Basic requirements for this project are as follows-

- Windows OS

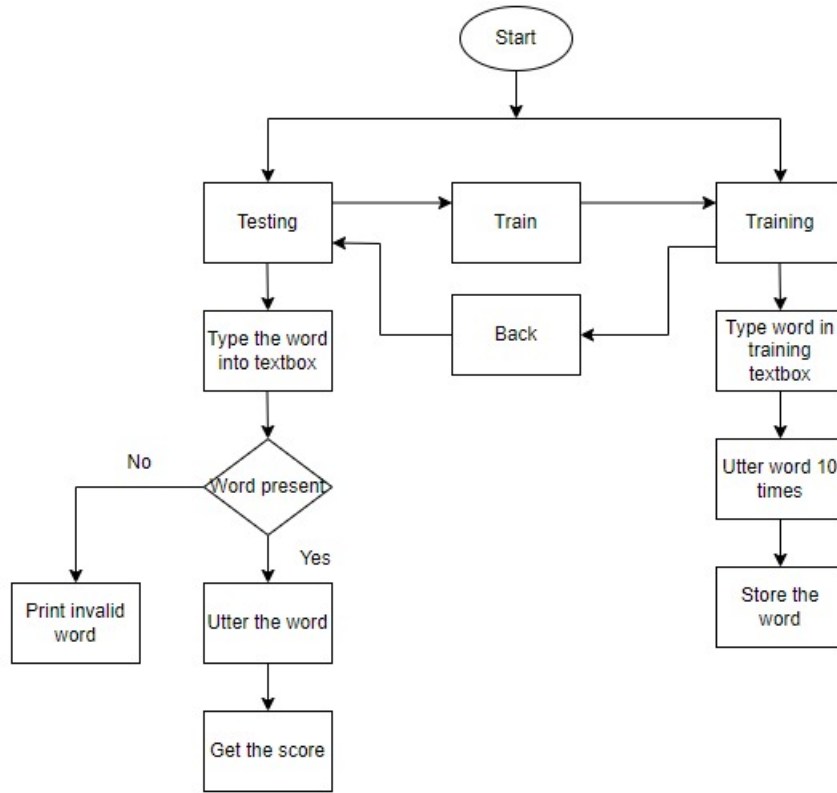


Figure 1: Flowchart of the project

- Microsoft Visual Studio 2010
- C++11 integrated with VS2010
- Recording Module
- A good microphone

4 Proposed Techniques

4.1 Flowchart

Figure 1 is a flowchart of the project. Those steps can be followed for successful execution of the project.

4.2 Model description

We are using the famous Hidden Markov Model to store the speech properties. Hidden Markov Model is a probabilistic model which is used to explain or derive the probabilistic characteristic of any random

process. When we apply log function to the spectral representation of the speech during reverse fourier transform, it is converted to cepstrum whose coefficients are steady because of application of log function and it represents the speech in a nice manner. This representation can be used as the speech property. We take all such cepstral coefficients and build a codebook which helps in generating the observation sequences. Codebook contains 50 speech samples for pre trained words and 10 for live testing.

We use feed-forward model for modelling speech samples. While speaking, we speak a word from start to end. So, there is no need of backward movement. Also, the stress on current phoneme is more than moving to the next phoneme. Hence we use feed-forward model. Then while testing, we score each model using the forward process and pick the word with highest score as the result.

Since, speech signals depend a lot on the environment, live testing might not be very good. But, if we train the model live and test it immediately, then we get significantly better accuracy.

5 Result

5.1 Home Page

Figure 2 shows the homepage of the project. It has a textbox to type the word to score and a score box to display the corresponding score. there is another button which allows to train new words for the application.

5.2 Live Scoring

In live scoring the word typed into the textbox is used to select the corresponding HMM model and to find the corresponding probability (Solution to the 1st problem of HMM). Figure 3 is the Live Scoring page .

5.3 Live Training

For live training, corresponding button should be clicked. Then Figure 4 will be displayed. A word can be entered. It will detect whether the word is already present. If the word is already present we will for the scoring procedure, otherwise recording will be done 20 times. For recording, the same Figure 6 will be displayed. After recording, Back button can be clicked to get back to homepage i.e. Figure 2.



Figure 2: Home Page

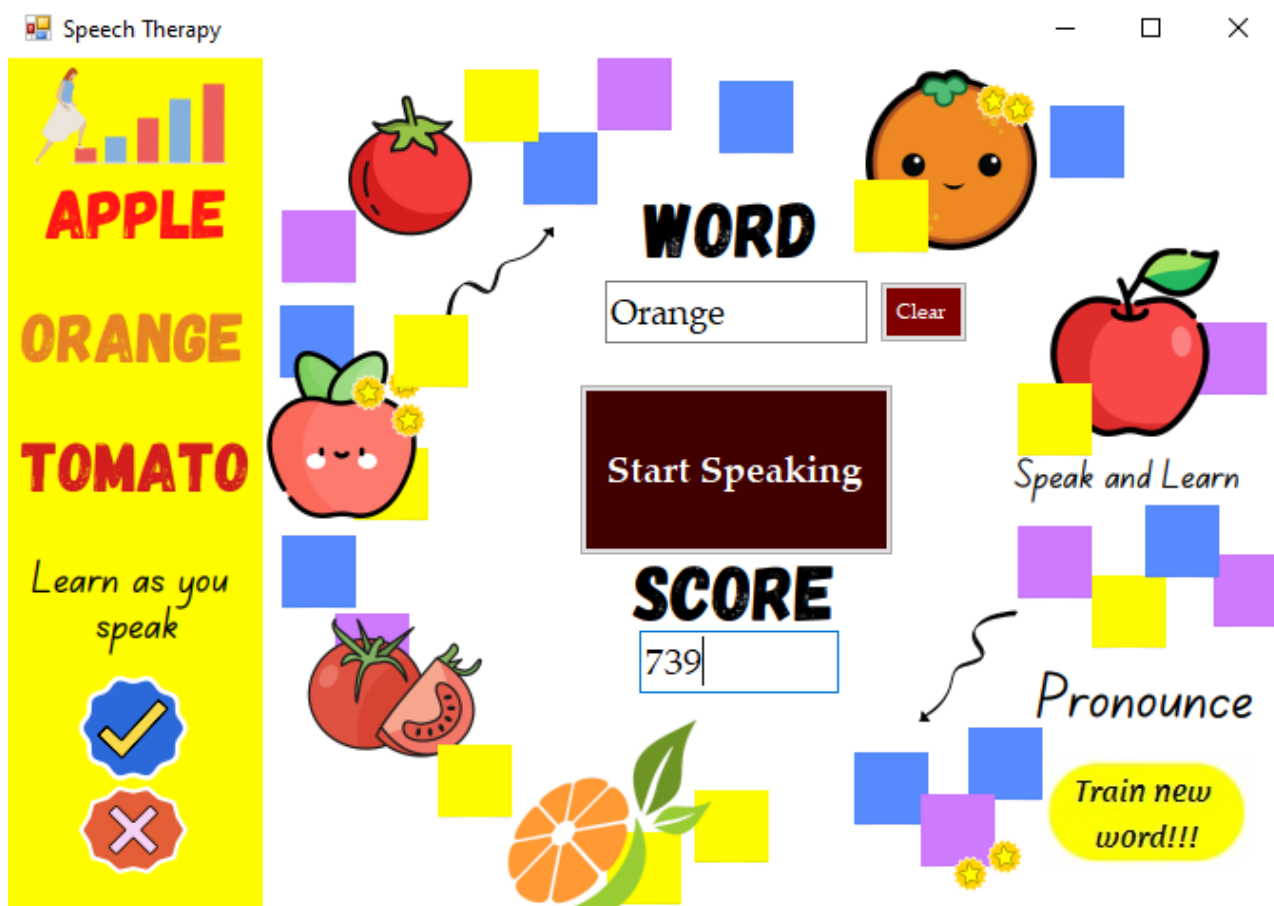


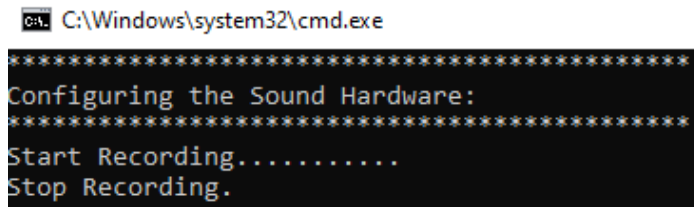
Figure 3: Live Scoring Section



Figure 4: Live Training



Figure 5: Live Training with new words

A screenshot of a Windows command prompt window. The title bar shows the icon for Command Prompt and the path 'C:\Windows\system32\cmd.exe'. The command prompt itself has a black background with yellow text. It displays a series of asterisks, followed by the text 'Configuring the Sound Hardware:', another series of asterisks, 'Start Recording.....', and 'Stop Recording.'. A white cursor is visible on the line following 'Stop Recording.'.

```
C:\Windows\system32\cmd.exe
*****
Configuring the Sound Hardware:
*****
Start Recording.....
Stop Recording.
_
```

Figure 6: Recording Console

6 Source Code

A few files are too large. Hence the entire code is not added here. A few short files are added.

6.1 test.h

```
// DigitRecognition.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
// DigitRecognition.cpp : Defines the entry point for the console application.
//

#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>
//!!!reading the input and turning it into ci values
///
///
#define NWORDS 100 //num of words
char WORDS[NWORDS][20]={ " apple", "orange", "tomato" };
//int NOFWords=3;
#define N 320
#define P 12

long double R[P+1]; // to calculate k[p] r[p] is needed
long double Kd[P+1]; //the K values for each iteration i 1 to p
long double prevAlpha[P+1]; // well at each time i of computation of both k(i) and curr
//value but an array of values, a whole table could be made tracking each alpha array fo
long double Alpha[P+1];// the current alpha array
long double prevE;// E value at i-1 iteration
long double currE;// E value at the current ith iteration
long double A[N]; // the 320 sample frame of amplitudes;
```

```

long double C[P+1];
long double Reference[5][P]; //5*12
long double fileInput[1000007];
long double tokuraW[P]={1.0, 3.0, 7.0, 13.0, 19.0, 22.0, 25.0, 33.0, 42.0, 50.0, 56.0, 6
int lowerB=7000;
int upperB=7000;

int M=0;//size of the universe dynamically seen

long double cepstralUniverse[100007][P];

void addWord(char* wordName)
{
    FILE* fp= fopen("wordCheck.txt","a+");
    fprintf(fp,"%s\n",wordName);
    fclose(fp);
}

long double summation(int i)
{
    long double sum=0.0;
    for(int j=1; j<=i-1;j++)
    {
        sum+=prevAlpha[j]* R[i-j];
    }
    return sum;
}

void durbins()
{
    ///lets initialize some base values
    prevE=R[0];
    //for p==1 things are a bit different
    Kd[1]= R[1]/R[0];
    Alpha[1]= Kd[1];

```

```

//rest of the current alphas for 1st iteration will remain 0, obviously they are
currE= (1- Kd[1]*Kd[1])*prevE;

//// hence the differences for the first iteration are calculated , many of these
//next iterations

for(int i=2;i<=P;i++)
{
    prevE=currE; // updating E for this iteration
    //first let us update alpha values , the current becomes previous to be
    for(int j=1;j<=i-1;j++) // previously we computer currPrev from 1 to i-1
    {
        prevAlpha[j] = Alpha[j];
    }
    //calculating the k for this iteration
    Kd[i]= (R[i]- summation(i))/prevE;
    //calculating the alphas of this iteration
    Alpha[i]=Kd[i];
    for(int j=1;j<=i-1;j++)
    {
        Alpha[j]= prevAlpha[j]- Kd[i]*prevAlpha[i-j];
    }
    //calculating E for this iteration
    currE= (1-Kd[i]*Kd[i])*prevE;
}

}

void computeRi()
{

```

```

    for (int i=0;i<=P;i++)
    {
        long double cal=0;
        for (int j=0;j<N-i;j++)
        {
            cal+= (A[j]*A[j+i]);
        }
        R[i]=cal;//divide by N not needed
    }
}

void computeCi()
{
    C[0]= log1(R[0]); //using energy for intial c0 value
    for (int i=1;i<=P;i++)
    {
        long double sumPart=0; //c[i]= a[i] + summation
        for (int k=1;k<=i-1;k++)
        {
            sumPart+= ((k)* C[k] * Alpha[i-k])/i; //i==n in the written form

        }
        C[i]= Alpha[i] + sumPart;
    }
}

void hammingWindow()
{
    for (int n=0;n<=N-1;n++)
    {
        long double x = (2*3.14159*n)/(N-1);
        long double Wn= 0.54 - (0.46*cos(x));
        A[n]= A[n] * Wn;
    }
}

```



```

}
void raisedSine()
{
    for(int m=1;m<=P;m++)
    {
        long double x = (3.14159*m)/P;
        long double Wm = 1 + (P/2)*sin(x);
        C[m]= C[m]*Wm;
    }
}

//!!!! CLAMPING PART ////
//calculating the threshold zcr and energy value

//return lower bound of clamp
int min(int a,int b)
{
    if(a<b)return a;
    else return b;
}
int max(int a,int b)
{
    if(a>b)return a;
    else return b;
}
long double energy(int sPos)
{
    long double ans=0.0;
    for(int i=sPos;i<sPos+N;i++)
    {
        ans+=(fileInput[i]*fileInput[i]);
    }
    return (ans/N);
}

```

```

}
void calculateBounds(int sampleSize)
{
    long double mx;
    int flag=1;
    int bestPos=3000;
    for (int i=1000;i<sampleSize-400;i+=N)
    {
        long double currE=energy(i);
        if(flag)
        {mx=currE; bestPos=i; flag=0;}
        else
        {
            if(currE>mx)
            {
                mx=currE;
                bestPos=i;
            }
        }

    }

    upperB= min(bestPos+3000,sampleSize-100);
    lowerB=max(bestPos-3000,0);

}

//!!!! CLAMPING PART ENDS ////
void readFile(char* fileName, int utterNo) // if nw is set then new word is being added
{
    //if utterNo ==0 then its the simple original one
    if(utterNo==1)M=0; //resets universe for the first utterance

    FILE* fp= fopen(fileName,"r");
    if(fp==NULL)

```

```

{
    return;
}
int j=0;
while(fscanf(fp,"%Lf",&fileInput[j]) > 0) //taking the amplitudes of utterance a
{
    j++;
}
//j is the no of sample in this
//this function give use the optimal clamping based on energy
calculateBounds(j);
printf("%d %d\n",lowerB,upperB);
for(int i=lowerB;i<(upperB);i+=80)
{
    int s=0;
    for(int l=i;l<i+N;l++)
    {
        A[s]=fileInput[l];
        s++;
    }
    computeRi();//compute ri for the current frame
    durbins();//compute ai values for the current frame
    computeCi();//compute ci values for the given frame
    raisedSine();//apply raised sine window
    //now c[1] — c[12] stores the cepstral vector
    //we add this to the universe

    for(int i=1;i<=12;i++)
    {
        cepstralUniverse[M][i-1]=C[i];
    }
    M++;//cuz 0 based indexing
}

```

```

        fclose(fp);

    }

    void read(int wordInd)
    {
        char pre[30]=" dataset //";

        char fileName[100]="";
        strcat(fileName,pre);
        char* s=WORDS[wordInd];
        addWord(s);
        strcat(fileName,s);
        char addMore[10]=" _";
        strcat(fileName,addMore);
        for(int j=1;j<=50;j++)
        {
            char res[10];
            char fname[100]="";

            sprintf(res,"%d",j);
            strcat(fname,fileName);
            strcat(fname,res);
            char res2[10]=".txt";
            strcat(fname,res2);
            readFile(fname,j);
        }

    }

    ///!! end of reading and converting to ci
    ///
    ///AFTER THIS SECTION I HAVE MY CEPSTRAL UNIVERSE ( ALL 0 BASED INDEXING TILL HMM)

```

```

////

//!!! MAKING THE CODEBOOK
//
//
#define K 32 //max size of the codebook
long double codebook[K][P];
long double tmpCodebook[K][P];

//takes in two cepstral vectors and returns the tokura distance between them
long double tokuraDistance(long double *c1, long double *c2)
{
    long double ans=0;
    for(int i=0;i<P;i++)
    {
        long double d= c1[i]-c2[i];
        ans= ans + (tokuraW[i]*(d*d));
    }
    return ans;
}

//picks random k vectors to initialize the codebook
void initCodebook()
{
    for(int i=0;i<K;i++)
    {
        int index= rand()%M;
        for(int j=0;j<P;j++)
        {
            codebook[i][j]=cepstralUniverse[index][j];
        }
    }
}

```

```

}
//compares tokura distance of c1 with centroid of each region/index in codebook
//and return index of the one with the lowest distance
int calcIndex(long double *c1,int currK)
{
    long double minD;//minimum distance
    int index;//index corresponding to region with the minD
    for(int i=0;i<currK;i++)
    {
        long double c2[P]; //this will store the centroid for the ith region in codebook
        for(int j=0;j<P;j++) c2[j]=codebook[i][j];
        if(i==0)
        {
            minD= tokuraDistance(c1,c2);
            index=i;
        }
        else
        {
            long double tmp= tokuraDistance(c1,c2);
            if(tmp<minD)
            {
                minD=tmp;
                index=i;
            }
        }
    }
    return index;
}
//initializes the temporary codebook
void initTmpCodebook()
{
    for(int i=0;i<K;i++)
    {
        for(int j=0;j<P;j++)

```

```

        {
            tmpCodebook[i][j]=0.0;
        }
    }

}

//loops across the universe of the cepstral vectors and assigns them to the tmpCodebook
//also updates the centroid in the codebook for the next iteration
//calculates the total distortion for this new codebook and returns it
long double calculateD(int currK)
{
    initTmpCodebook();
    int cntRegion[K]; //stores no of vectors assigned to each region in this iteration
    for(int i=0;i<K;i++) cntRegion[i]=0;
    for(int i=0;i<M;i++)
    {
        long double c1[P]; //this will store a c vec from universe

        for(int j=0;j<P;j++)
        {
            c1[j]=cepstralUniverse[i][j];
        }
        int index= calcIndex(c1,currK); //index where this vector should be added
        //printf("%d\n",index);
        //in tmp codebook
        //so i assign c1 to index index in tmp codebook( additive assign)
        cntRegion[index]++;
        for(int j=0;j<P;j++)
        {
            tmpCodebook[index][j]+= c1[j];
        }
    }
}

//now we average out the tmpCodebook and assign to codebook
//careful not to do div by 0( in case no vector assigned to a particular index)

```

```

//that shouldnt happen because the random vector we assigned would atleast be assigned
for(int i=0;i<currK;i++)
{
    for(int j=0;j<P;j++)
    {
        if(cntRegion[i]==0)
        {
            printf("fuck\n");
            continue;
        }
        codebook[i][j]= tmpCodebook[i][j]/cntRegion[i];
    }
}
//now the centroid in the codebook are updated
//lets calculate distortion of the current codebook and return it
long double d=0.0; //distortion of current codebook
for(int i=0;i<M;i++)
{
    long double c1[P]; //this will store a c vec from universe
    for(int j=0;j<P;j++)
    {
        c1[j]=cepstralUniverse[i][j];
    }
    int index=calcIndex(c1,currK);
    long double c2[P]; //this is the centroid for this index
    for(int j=0;j<P;j++)
    {
        c2[j]=codebook[index][j];
    }
    d+= tokuraDistance(c1,c2);
}
return d/M;
}
//takes input

```



```

//returns abs val of diff of d1 and d2
long double absD(long double d1,long double d2)
{
    long double ans= d2-d1;
    if(ans<0.0) return -1*ans;
    return ans;
}
//calculates k means on the curr size of the codebook = currK
void kMeans(int currK)
{

    long double d1= calculateD(currK);
    long double d2= calculateD(currK);
    long double deltaCurr= absD(d1,d2);
    int itr=0;
    while(deltaCurr>0.00001)
    {
        d1=d2;
        d2=calculateD(currK);
        deltaCurr=absD(d1,d2);
        itr++;
    }
    printf("%d\n", itr);
}
//split the codebook
void split(int currK)
{
    long double eps=0.03;
    //lets make a tmp codebook equal to curr codebook and then we update in curr codebook
    //will have c+e, c-e at 2i and 2i+1 indexes
    long double tmpCb[K][P];
    for(int i=0;i<K;i++)
    {

```

```

        for (int j=0;j<P;j++)
        {
            tmpCb[i][j]=codebook[i][j];
        }
    }
    for (int i=0;i<currK;i++)
    {
        for (int j=0;j<P;j++)
        {
            codebook[2*i][j]= tmpCb[i][j]+eps;
            codebook[2*i+1][j]=tmpCb[i][j]-eps;
        }
    }
}

//the lbg algorithm , we build codebook iteratively by splitting till size ==K=8 of codebook
void lbg()
{

    initCodebook();//initializes codebook of size 1,randomly assigns 1 vector
    int currK; //current codebook size

    for (currK=1;currK<=K;currK*=2)
    {
        kMeans(currK);//run k means for currK size of codebook
        split(currK); //now we split the codebook

    }
    printf("\ndone");
}

void makeCodebook()
{
    lbg();
}

```

```

void storeCodebook(int wordInd)
{
    //stores in codebooks//
    char *wordName= WORDS[wordInd];
    char pre[20]= "codebooks//";
    char lst[20]=".txt";
    char fName[100]="";
    strcat(fName,pre);
    strcat(fName,wordName);
    strcat(fName,lst);
    FILE* fp= fopen(fName,"w");
    for(int i=0;i<32;i++)
    {
        for(int j=0;j<12;j++)
        {
            fprintf(fp,"%Lf ",codebook[i][j]);
        }
        fprintf(fp,"\n");
    }
    fclose(fp);
}

//

//now we train our hmm model
//we will train 10 models
//for each model we have 25 utterances to train with
//to train 1 model, we take each of 25 utterance convert to observation seq and avg out

//////////
#define T 300 //just a upperbound on no of observations, otherwise we dynamically take i
int OSeqSz[20][70]; //stores size of the observsation seq
int OSeq[20][70][T]; //20 words 70 utterances
//takes in a file name and creates observation sequence for it

```

```

//also takes in x which is word no and y which is utterance no
//basically puts in [x][y]

void makeOSeq(char* fileName,int x,int y)
{
    FILE* fp= fopen(fileName,"r");
    if(fp==NULL)
    {
        return;
    }
    int j=0;
    while(fscanf(fp,"%Lf",&fileInput[j]) > 0) //taking the amplitudes of utterance a
    {
        j++;
    }
    //j is the no of sample in this
    //!!! should i include the way of cutting the silence parts?, just ignore first
    int oSize=0; //calculates size of curr o seq
    int oIndex=0;
    calculateBounds(j);

    for(int i=lowerB;i<(upperB);i+=80)
    {

        int s=0;
        for(int l=i;l<i+N;l++)
        {
            A[s]=fileInput[l];
            s++;
        }
        computeRi();//compute ri for the current frame
        durbins();//compute ai values for the current frame
        computeCi();//compute ci values for the given frame
        raisedSine();//apply raised sine window
    }
}

```

```

//now c[1] — c[12] stores the cepstral vector
//we add this to the universe

int flag=1;
long double mm;
int ansInd; //index of codebook which is closest to current vector
long double ccurr[P];
for(int lmao=0;lmao<P;lmao++)
{
    ccurr[lmao]=C[lmao+1];
}
for(int lm=0;lm<K;lm++)
{
    long double cind[P];
    for(int lma=0;lma<P;lma++)
    {
        cind[lma]=codebook[lm][lma];
    }

    if(flag)
    {
        flag=0;
        mm=tokuraDistance(ccurr,cind);
        ansInd=lm;
    }
    else
    {
        long double valt= tokuraDistance(ccurr,cind);
        if(valt<mm){mm=valt; ansInd=lm;}
    }
}
OSeq[x][y][oIndex++]=ansInd;
oSize++;
}

```

```

        OSeqSz[x][y]=oSize;

    }
    //find and store observation seq for all the training data
    void allOSeq(int wordInd)
    {
        char pre[30]=" dataset //";
        char fileName[100]="";
        strcat(fileName,pre);
        char* s=WORDS[wordInd];
        strcat(fileName,s);

        char addMore[10]=" _";
        strcat(fileName,addMore);
        for(int j=1;j<=50;j++)
        {
            char res[10];
            char fname[100]="";

            sprintf(res,"%d",j);
            strcat(fname,fileName);
            strcat(fname,res);
            char res2[10]=".txt";
            strcat(fname,res2);
            makeOSeq(fname,wordInd,j-1);
        }
    }
    ///hmmmm for 1 utterance of 1 digit

#define N 5
#define T 300
    long double Ai[N+1][N+1];
    long double B[N+1][K+1];
    long double alpha[T+1][N+1];

```

```

int observationSequence [T+1];
long double pi [N+1];
long double beta [T+1][N+1];
long double delta [T+1][N+1];
int psi [T+1][N+1];
long double Pstar;
int Qstar [T+1];
long double Gamma [T+1][N+1];
long double zeeta [T+1][N+1][N+1];
#define threshold 1e-30
int currOSize=60;

```

```

void readA()
{
    FILE* fp= fopen("A.txt","r");

    if (fp==NULL)
    {
        printf("error reading\n");
        return;
    }
    long double x;
    for (int i=1;i<=N;i++)
    {
        for (int j=1;j<=N;j++)
        {
            fscanf(fp,"%Lf",&x);
            Ai[i][j]=x;
        }
    }
    fclose(fp);
}

```

```

}
void readB()
{
    FILE* fp= fopen("B.txt","r");

    if(fp==NULL)
    {
        printf("error reading\n");
        return;
    }
    long double x;
    for(int i=1;i<=N;i++)
    {
        for(int j=1;j<=K;j++)
        {
            fscanf(fp,"%Lf",&x);
            B[i][j]=x;
        }
    }
    fclose(fp);

}

//reads the stored o seq of the xth digit and its yth utterance
//int OSeqSz[10][25]; //stores size of the observsation seq
//int OSeq[10][25][T]; //10 digits 25 utterances each
void readObservationSequence(int x,int y)
{
    currOSize=OSeqSz[x][y];
    for(int i=1;i<=OSeqSz[x][y];i++)
    {
        observationSequence[i]=OSeq[x][y][i-1];
    }
}

void readPi()

```



```

{
    FILE* fp= fopen("pi.txt","r");
    if (fp==NULL)
    {
        printf("error reading\n");
        return;
    }
    long double x;
    for (int i=1;i<=N;i++)
    {
        fscanf(fp,"%Lf",&x);
        pi[i]=x;
    }
    fclose(fp);
}

///

```

```

        {
            accum+= (alpha[t][i]*Ai[i][j]);
        }
        alpha[t+1][j]= accum*B[j][observationSequence[t+1]];
    }

}

}

long double terminationFWD()
{
    long double accum=0.0;
    for(int i=1;i<=N;i++)
    {
        accum+= alpha[currOSize][i];
    }
    return accum;
}

long double fwdProcedure()
{
    initializeFWD();
    inductionFWD();
    long double score = terminationFWD();
    printf("The score is : %Le\n",score);
    return score;
}

}

//!! END OF FWD PROCEDURE
///BWD PROCEDURE START
void initBWD()
{
    for(int i=1;i<=N;i++)

```

```

        {
            beta[currOSize][i]=1;
        }
    }
void inductionBWD()
{
    for(int t=currOSize-1;t>=1;t--)
    {
        for(int i=1;i<=N;i++)
        {
            long double accum=0.0;
            for(int j=1;j<=N;j++)
            {
                accum+= (Ai[i][j]*B[j][observationSequence[t+1]]*beta[t+1][j]);
            }
            beta[t][i]= accum;
        }
    }
}
void bwdProcedure()
{
    initBWD();
    inductionBWD();
}

//!!!BWD Procedure ends

///// VITERBI Starts
void initViterbi()
{
    for(int i=1;i<=N;i++)
    {
        delta[1][i]= pi[i]*B[i][observationSequence[1]];
        psi[1][i]=0;
    }
}

```

```

    }

}

void viterbiInduction()
{
    for(int t=2;t<=currOSize;t++)
    {
        for(int j=1;j<=N;j++)
        {
            //delta computation
            //find max val over i (1 to N)
            long double mx= delta[t-1][1]*Ai[1][j];
            int mxIndex=1;
            for(int i=1;i<=N;i++)
            {
                long double val = delta[t-1][i]*Ai[i][j];
                if(val>mx){mx=val;mxIndex=i;}
            }
            delta[t][j]= mx*B[j][observationSequence[t]];

            //psi computation
            psi[t][j]=mxIndex;

        }
    }
}

void viterbiTermination()
{
    //compute pstar
    long double mx= delta[currOSize][1];
    int mxIndex=1;
    for(int i=1;i<=N;i++)
    {
        long double val= delta[currOSize][i];
    }
}

```

```

        if (val>mx)
        {
            mx=val;
            mxIndex=i;
        }
    }
    Pstar=mx;
    Qstar[currOSize]=mxIndex;

}

void viterbiBackTrack()
{
    for (int t=currOSize-1;t>=1;t--)
    {
        Qstar[t]= psi[t+1][Qstar[t+1]];
    }
}

void viterbi()
{
    initViterbi();
    viterbiInduction();
    viterbiTermination();
    viterbiBackTrack();
    printf("pstar val is : %Le\n",Pstar);

}

/////viterbi ends
/////buam welch begins
void populateGamma()
{
    for (int t = 1; t <= currOSize; t++)

```

```

{
    long double accum=0;
    for (int i = 1; i <= N; i++)
    {
        accum += (alpha[t][i] * beta[t][i]);
    }
    for (int i = 1; i <= N; i++)
    {
        if(accum!=0)
            Gamma[t][i] = (alpha[t][i] * beta[t][i]) /accum;
        else Gamma[t][i]=0;
    }
}

}

void zeetaMatrix()
{
    for (int t = 1; t <= currOSize - 1; t++)
    {
        long double denom = 0;
        for (int i = 1; i <= N; i++)
        {
            for (int j = 1; j <= N; j++)
            {
                denom += (alpha[t][i] * Ai[i][j] * B[j][observationSequence[t + 1]] * be

            }
        }

        for (int i = 1; i <= N; i++)
        {
            for (int j = 1; j <= N; j++)
            {
                if(denom!=0)
                    zeeta[t][i][j] = (alpha[t][i] * Ai[i][j] * B[j][observationSequence[t +

```

```

else zeeta[t][i][j]=0;

    }

}

}

void reEstimatePi()
{
    for (int i = 1; i <= N; i++)
    {
        pi[i] = Gamma[1][i];
    }
}

void reEstimateA()
{
    for (int i = 1; i <= N; i++)
    {
        for (int j = 1; j <= N; j++)
        {
            long double numAccum = 0;
            long double denomAccum = 0;

            for (int t = 1; t <= currOSize - 1; t++)
            {
                numAccum += zeeta[t][i][j];
                denomAccum += Gamma[t][i];
            }

            if (denomAccum!=0)
                Ai[i][j] = (numAccum / denomAccum);
            else Ai[i][j]=0;
        }
    }
}

```

```

void reEstimateB()
{
    for (int j = 1; j <= N; j++)
    {
        for (int k = 1; k <= K; k++)
        {
            long double numAccum = 0;
            long double denomAccum = 0;
            for (int t = 1; t <= currOSize; t++)
            {
                if (observationSequence[t] == k && Qstar[t] == j )
                {
                    numAccum += Gamma[t][j];
                }
                denomAccum += Gamma[t][j];
            }

            if (denomAccum!=0)
                B[j][k] = numAccum / denomAccum;
            else B[j][k]=threshold;
            if (B[j][k] == 0)
            {
                B[j][k] = threshold;
            }
        }
    }
}

void buamWelch()
{
    populateGamma();
    zeetaMatrix();
}

```



```

        reEstimatePi();
        reEstimateA();
        reEstimateB();
    }
    ///buam welch ends
    ///
    //creates a model lambda to fit yth utterance of xth digit
    void modelXY(int x , int y,int itr)
    {
        if(itr==0)readA();
        if(itr==0)readB();
        readObservationSequence(x,y);
        if(itr==0)readPi();
        for(int i=0;i<20;i++)
        {
            fwdProcedure();
            bwdProcedure();
            viterbi();
            buamWelch();
            printf("end of 1 iter\n");
        }
    }

    ///
    //creating the 10 lambdas by running hmm on each utterance and avg out for each digit 3
    //long double Ai[N+1][N+1];
    //long double B[N+1][K+1];
    //long double pi[N+1]
    long double avgAi[10][N+1][N+1]; //stores the averageof the word models
    long double avgB[10][N+1][K+1];
    long double avgPi[10][N+1];
    void addA(int index)

```

```

{
    for (int i=1;i<=N;i++)
    {
        for (int j=1;j<=N;j++)
            avgAi[index][i][j]+=Ai[i][j];
    }
}

void addB(int index)
{
    for (int i=1;i<=N;i++)
    {
        for (int j=1;j<=K;j++)
            avgB[index][i][j]+=B[i][j];
    }
}

void addPi(int index)
{
    for (int i=1;i<=N;i++) avgPi[index][i]+=pi[i];
}

void avgitA(int index)
{
    for (int i=1;i<=N;i++)
    {
        for (int j=1;j<=N;j++)
            avgAi[index][i][j]/=25;
    }
}

void avgitB(int index)
{
    for (int i=1;i<=N;i++)
    {
        for (int j=1;j<=K;j++)
            avgB[index][i][j]/=25;
    }
}

```

```

}
void avgitPi(int index)
{
    for(int i=1;i<=N;i++) avgPi[index][i]/=25;
}
void readAPrev(int index)
{
    for(int i=1;i<=N;i++)
    {
        for(int j=1;j<=N;j++)
            Ai[i][j]=avgAi[index][i][j];
    }
}

void readBPrev(int index)
{
    for(int i=1;i<=N;i++)
    {
        for(int j=1;j<=K;j++)
            B[i][j]=avgB[index][i][j];
    }
}

void readPiPrev(int index)
{
    for(int i=1;i<=N;i++) pi[i]=avgPi[index][i];
}

///// making stored mode stochastic
void ensureStochastic()
{
    long double rowMax = 0, rowSum = 0, normalizer = 0;
    int maxIndex = 0;
    int i = 0, j = 0;

```

```

for (i = 1; i <= 5; ++i)
{
    rowMax = rowSum = 0;
    maxIndex = -1;
    for (j = 1; j <= 5; ++j)
    {
        if (Ai[i][j] > rowMax)
        {
            rowMax = Ai[i][j];
            maxIndex = j;
        }
        rowSum += Ai[i][j];
    }
    if (rowSum != 1)
    {
        normalizer = abs(rowSum-1);
        Ai[i][maxIndex] = rowSum > 1 ? Ai[i][maxIndex]-normalizer : Ai[i]
    }
}

for (i = 1; i <= 5; ++i)
{
    rowMax = rowSum = 0;
    maxIndex = -1;
    for (j = 1; j <= 32; ++j)
    {
        if (B[i][j] > rowMax)
        {
            rowMax = B[i][j];
            maxIndex = j;
        }
        rowSum += B[i][j];
    }
    if (rowSum != 1)

```

```

        {
            normalizer = abs(rowSum-1);
            B[i][maxIndex] = rowSum > 1 ? B[i][maxIndex]-normalizer : B[i][maxIndex];
        }
    }
}

void makeStochastic(char *wordName)
{
    char pre[20]= "models/";
    char lst[20]=".txt";
    char fName[100]="";
    strcat(fName,pre);
    strcat(fName,wordName);
    strcat(fName,lst);
    FILE* fp= fopen(fName,"r");
    if(fp==NULL) return;
    for(int i=1;i<=N;i++)
    {
        for(int j=1;j<=N;j++)
        {
            fscanf(fp,"%Le",&Ai[i][j]);
        }
    }
    //fscanf(fp,"%c",&x);
    for(int i=1;i<=N;i++)
    {
        for(int j=1;j<=K;j++)
        {
            fscanf(fp,"%Le",&B[i][j]);
        }
    }
    //fscanf(fp,"%c",&x);
    for(int i=1;i<=N;i++)

```

```

{
    fscanf(fp,"%Le",&pi[i]);
}
ensureStochastic();
fclose(fp);
FILE* fp1= fopen(fName,"w");
if(fp1==NULL) return;
for(int i=1;i<=N;i++)
{
    for(int j=1;j<=N;j++)
    {
        fprintf(fp1,"%Le",Ai[i][j]);
    }
    fprintf(fp1,"\n");
}
//fscanf(fp,"%c",&x);
for(int i=1;i<=N;i++)
{
    for(int j=1;j<=K;j++)
    {
        fprintf(fp1,"%Le",B[i][j]);
    }
    fprintf(fp1,"\n");
}
//fscanf(fp,"%c",&x);
for(int i=1;i<=N;i++)
{
    fprintf(fp1,"%Le",pi[i]);
}
fclose(fp1);
}
///end of stochastic make

```

```

void storeModel(int wordInd)
{

    char *wordName= WORDS[wordInd];
    char pre[20]= "models//";
    char lst[20]=".txt";
    char fName[100]="";
    strcat(fName,pre);
    strcat(fName,wordName);
    strcat(fName,lst);
    FILE* fp= fopen(fName,"w");
    for(int x=1;x<=N;x++)
    {
        for(int y=1;y<=N;y++)
        {
            fprintf(fp,"%Le ",avgAi[wordInd][x][y]);
        }
        fprintf(fp,"\n");
    }
    for(int x=1;x<=N;x++)
    {
        for(int y=1;y<=K;y++)
        {
            fprintf(fp,"%Le ",avgB[wordInd][x][y]);
        }
        fprintf(fp,"\n");
    }
    for(int x=1;x<=N;x++)
    {
        fprintf(fp,"%Le ",avgPi[wordInd][x]);
    }
    fclose(fp);
}

```

```

}
void hmmModel(int wordInd)
{
//models are 0 based index and utterances are 0 based stored in data structures
    for(int itr=0;itr<3;itr++)
    {
        for(int i=wordInd;i<=wordInd;i++)//ith word
        {
            for(int j=0;j<50;j++)
            {
                if(itr>0) //read previous model
                {
                    readAPrev(i);
                    readBPrev(i);
                    readPiPrev(i);
                }
                modelXY(i,j,itr);
                //now lets add the 3 models to their respective averages
                addA(i);
                addB(i);
                addPi(i);
            }
            avgitA(i);
            avgitB(i);
            avgitPi(i);
        }

    }

    storeModel(wordInd);//stores all the word models in file
    //making stored model stochastic
    char *s= WORDS[wordInd];
    makeStochastic(s);
}

```



```

}

/////
//int OSeqSz[10][30]; // stores size of the observsation seq
//int OSeq[10][30][T]; //10 digits 30 utterances each
/////
void loadA(int index)
{
    for(int i=1;i<=N;i++)
    {
        for(int j=1;j<=N;j++)
            Ai[i][j]=avgAi[index][i][j];
    }
}

void loadB(int index)
{
    for(int i=1;i<=N;i++)
    {
        for(int j=1;j<=K;j++)
            B[i][j]=avgB[index][i][j];
    }
}

void loadPi(int index)
{
    for(int i=1;i<=N;i++)
    {
        pi[i]= avgPi[index][i];
    }
}

long double scoreWith(int k)//calculate score of the loades o seq against the kth model

```

```

{
    if (k!=7)
    {
        loadA(k);
        loadB(k);
        loadPi(k);
    }
    return fwdProcedure();
}

void testing()
{
    printf("\n\nTESTING!!!!!!!!\n\n\n !!!!\n\n");
    //convert the 5 utterances of test to o seq and use pb1 to score against a model
    long double totalAccuracy=0.0;
    for(int i=0;i<NWORDS;i++)

    {

        int hits=0;
        for(int j=41;j<=45;j++)
        {
            readObservationSequence(i,j);
            printf("word %d utterance %d\n",i,j);
            //reads the observation sequence
            //now we iterate over the 10 models
            //calculate scores for this loaded o seq and return the one with
            long double score=50;
            int flag=1;
            int bestM=0; //index of best model
            for(int k=0;k<NWORDS;k++) //testing against kth model
            {
                long double tmp = scoreWith(k);
                if(tmp!=0 && score!=50)

```

```

        { if (tmp>score){ score=tmp; bestM=k;}}
        else if (tmp!=0 && score==50) {bestM=k; score=tmp

    }
    if (bestM==i) hits++;
    printf("best score is : %Le and predicon is word : %d\n",score ,

}
long double accuracy= (long double)hits/5.0;
printf("%Lf\n",accuracy);
totalAccuracy+=accuracy;

}
totalAccuracy/=NWORDS;
printf("%Lf\n",totalAccuracy);
}

void printModel()
{
    for (int k=0;k<10;k++)
    {
        printf("\n\nanother model final!!!\n\n");
        for (int i=1;i<=N;i++)
        {
            for (int j=1;j<=N;j++)
            {
                printf("%Lf ",avgAi[k][i][j]);
            }
            printf("\n\n");
        }
        for (int i=1;i<=N;i++)
        {
            for (int j=1;j<=K;j++)
            {
                printf("%Lf ",avgB[k][i][j]);
            }
            printf("\n\n");
        }
    }
}

```

```

    }

}

}

/////
/*
//to dynamically take i gotta store that as well
int OSeqSz[10][30]; //stores size of the observsation seq
int OSeq[10][30][T]; //10 digits 30 utterances each
//each digit has 30 observatin seq
*/
void printOseq()
{
    for(int i=0;i<NWORDS;i++)
    {
        for(int j=0;j<50;j++)
        {
            printf("THis is word %d and utterance %d with size %d : \n",i,j,

            for(int k=0;k<100;k++)
            {
                printf("%d ",OSeq[i][j][k]);
            }
            printf("\n");
        }
    }
}

void loadModel(int wordInd)
{
    char *wordName= WORDS[wordInd];
    char pre[20]= "models//";

```

```

char lst[20]=".txt";
char fName[100]="";
strcat(fName,pre);
strcat(fName,wordName);
strcat(fName,lst);
FILE* fp= fopen(fName,"r");
if(fp==NULL) return;
for(int i=1;i<=N;i++)
{
    for(int j=1;j<=N;j++)
    {
        fscanf(fp,"%Le",&Ai[i][j]);
    }
}
//fscanf(fp,"%c",&x);
for(int i=1;i<=N;i++)
{
    for(int j=1;j<=K;j++)
    {
        fscanf(fp,"%Le",&B[i][j]);
    }
}
//fscanf(fp,"%c",&x);
for(int i=1;i<=N;i++)
{
    fscanf(fp,"%Le",&pi[i]);
}

fclose(fp);
}
void loadCodebook(int wordInd)
{

```

```

        char *wordName= WORDS[wordInd];
char pre[20]= "codebooks//";
char lst[20]=".txt";
char fName[100]="";
strcat(fName,pre);
strcat(fName,wordName);
strcat(fName,lst);
FILE* fp= fopen(fName,"r");
    if(fp==NULL) return;
    for(int i=0;i<32;i++)
    {
        for(int j=0;j<12;j++)
        {
            fscanf(fp,"%Lf",&codebook[i][j]);
        }
    }
    fclose(fp);
}
long double liveTesting()
{
    system("Recording_Module.exe 1 input.wav input.txt");
    char fName[]="input.txt";
    FILE *fp= fopen(fName,"r");
    if(fp==NULL) return 0.0;
    makeOSeq(fName,7,55);// these indexes store the live test oseq
    readObservationSequence(7,55);
    long double score = scoreWith(7); //7 means it takes loadmodel value to score with
    //scale the score?
    FILE* fp1= fopen("lmao.txt","a+");
    fprintf(fp1,"%Le",score);
    fclose(fp1);
}

```

```

        return score;
    }
int scaling_function(long double value)
{
    long double fraction;
    int e;

    if(value ==0) return 0;

    /*
    The range of long double is 4.94e-324 to 1.79e+308 on the positive side
    frexp returns powers in terms of 2 and the range is -1073 to 1024

    */

    fraction = frexp(value, &e);

    /*
    Since we are scaling probabilities the range would be from 1e+0 to 4.94e-324

    so power of 2 would range from 0 to -1073

    so subtracting the power from 1073 would be good base for scoring.
    */

    int score = (1073 - abs(e));

    if (score < 1000)
    {
        return score;
    }
    else
    {
        while (score > 1000)

```

```

        score = score - 10;

    return score;
}
}

void loadMAgain(char *wordName)
{

    char pre[20]= "models//";
    char lst[20]=".txt";
    char fName[100]="";
    strcat(fName,pre);
    strcat(fName,wordName);
    strcat(fName,lst);
    FILE* fp= fopen(fName,"r");
    if(fp==NULL) return;
    for(int i=1;i<=N;i++)
    {
        for(int j=1;j<=N;j++)
        {
            fscanf(fp,"%Le",&Ai[i][j]);
        }
    }
    //fscanf(fp,"%c",&x);
    for(int i=1;i<=N;i++)
    {
        for(int j=1;j<=K;j++)
        {
            fscanf(fp,"%Le",&B[i][j]);
        }
    }
    //fscanf(fp,"%c",&x);
    for(int i=1;i<=N;i++)

```



```

        {
            fscanf(fp,"%Le",&pi[i]);
        }

        fclose(fp);

    }
void loadCodebookAgain(char *wordName)
{
    char pre[20]= "codebooks//";
    char lst[20]=".txt";
    char fName[100]="";
    strcat(fName,pre);
    strcat(fName,wordName);
    strcat(fName,lst);
    FILE* fp= fopen(fName,"r");
    if(fp==NULL) return;
    for(int i=0;i<32;i++)
    {
        for(int j=0;j<12;j++)
        {
            fscanf(fp,"%Lf",&codebook[i][j]);
        }
    }
    fclose(fp);
}
int lTest(char* wordName)
{
    loadMAgain(wordName);
    loadCodebookAgain(wordName);
    long double score = liveTesting();
}

```

```

        return scaling_function(score);
    }
    /////// adding of new words
    //mainly i need to make a new codebook
    //the codebook will be for the specific word added now
    //i make the universe for this now
    //flag is 1 for first utterance of a word
    void addUtterance(char* wordName,int utterNo)
    {
        system("Recording_Module.exe 1 input.wav input.txt");
        char fName[20]="input.txt";
        //here we copy the input.txt with proper labels for future storage
        char thisFile[100]="nwWords/";
        strcat(thisFile,wordName);
        char uttNo[10];
        sprintf(uttNo,"%d",utterNo);
        strcat(thisFile,uttNo);
        char aur[10]=".txt";
        strcat(thisFile,aur);
        FILE* fp1= fopen(thisFile,"w");
        FILE* fp= fopen(fName,"r");
        long double x;
        while(fscanf(fp,"%Lf",&x) > 0) //taking the amplitudes of utterance as input
        {
            fprintf(fp1,"%Lf\n",x);
        }
        fclose(fp);
        fclose(fp1);

        //
        readFile(fName,utterNo); //this will read the input file , convert that to cepstr
        //now we use this file to create a codebook
    }

```

```

}
//void makeOSeq(char* fileName,int x,int y)

void makeOSeqNwWord(char *wordName,int wordInd)
{
    for(int i=1;i<=20;i++)
    {
        //wordInd indexed word and i th utterance of that, wordInd is 0 based and
        //to makeOSeq i pass wordName+i ,wordind,i
        char thisFile[100]="nwWords//";
        strcat(thisFile,wordName);
        char uttNo[10];
        sprintf(uttNo,"%d",i);
        strcat(thisFile,uttNo);
        char aur[10]=".txt";
        strcat(thisFile,aur);
        makeOSeq(thisFile,wordInd,i);
    }
}

void saveCurrModel(int wordInd,char *wordName)
{
    char thisFile[100]="models//";
    strcat(thisFile,wordName);
    char aur[10]=".txt";
    strcat(thisFile,aur);
    FILE* fp= fopen(thisFile,"w");
    for(int x=1;x<=N;x++)
    {
        for(int y=1;y<=N;y++)
        {

```

```

        fprintf(fp,"%Lf ",avgAi[wordInd][x][y]);
    }
    fprintf(fp,"\n");
}
for (int x=1;x<=N;x++)
{
    for (int y=1;y<=K;y++)
    {
        fprintf(fp,"%Lf ",avgB[wordInd][x][y]);
    }
    fprintf(fp,"\n");
}
for (int x=1;x<=N;x++)
{
    fprintf(fp,"%Lf ",avgPi[wordInd][x]);
}
fclose(fp);

}

```

```

void storeCodebookNw(char* wordName)
{
    //stores in codebooks//
    char pre[20]= "codebooks//";
    char lst[20]=".txt";
    char fName[100]="";
    strcat(fName,pre);
    strcat(fName,wordName);
    strcat(fName,lst);
    FILE* fp= fopen(fName,"w");
    for (int i=0;i<32;i++)
    {
        for (int j=0;j<12;j++)

```

```

        {
            fprintf(fp,"%Lf ",codebook[i][j]);
        }
        fprintf(fp,"\n");
    }
    fclose(fp);
}

int addNewWord(char *wordName,int wordInd)
{
    addWord(wordName); //added for checking purpose

    for(int i=1;i<=20;i++)
    {
        addUtterance(wordName,i); //adds 1 utterance
    }

    makeCodebook();
    storeCodebookNw(wordName);
    makeOSeqNwWord(wordName,wordInd);

    for(itr=0;itr<3;itr++)
    {

        for(int i=1;i<=20;i++)
        {
            if(itr>0) //read previous model
            {
                readAPrev(i);
                readBPrev(i);
                readPiPrev(i);
            }

            modelXY(wordInd,i,itr);
        }
    }
}

```

```

        addA(wordInd);
        addB(wordInd);
        addPi(wordInd);

    }
    avgitA(wordInd);
    avgitB(wordInd);
    avgitPi(wordInd);

}

saveCurrModel(wordInd,wordName);
makeStochastic(wordName);
return 1;
}

void trainWord(int wordInd)
{
    //trains word at wordInd of the pre recorded words
    read(wordInd);
    makeCodebook();
    storeCodebook(wordInd);
    allOSeq(wordInd);
    hmmModel(wordInd);

}
void trainWords()
{
    for(int i=0;i<3;i++) trainWord(i);
}

int checkWord(char* wordName)

```

```

{
    //we go through file and check for words if present or not
    FILE* fp = fopen("wordCheck.txt","r");
    char words[100];
    while( fscanf(fp,"%s\n",words)>0)
    {
        if(strcmp(words,wordName)==0)return 1;
    }

    fclose(fp);
    return 0;
}

```

```

/////

```

```

%

```

6.2 Form1.h

```

#pragma once
#include "test.h"
int word_number = 3;
namespace Neha_interface {

    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;

    /// <summary>

```

```

/// Summary for Form1
/// </summary>
public ref class Form1 : public System::Windows::Forms::Form
{
public:
    Form1(void)
    {
        InitializeComponent();
        //
        //TODO: Add the constructor code here
        //
    }

protected:
    /// <summary>
    /// Clean up any resources being used.
    /// </summary>
    ~Form1()
    {
        if (components)
        {
            delete components;
        }
    }

private: System::Windows::Forms::TextBox^ textBox1;
private: System::Windows::Forms::Button^ button1;

private: System::Windows::Forms::TextBox^ textBox3;

private: System::Windows::Forms::Label^ label4;

private: System::Windows::Forms::Button^ button2;

```



```

private: System::Windows::Forms::Button^    button4;
private: System::Windows::Forms::TextBox^    textBox2;
private: System::Windows::Forms::Button^    button5;
private: System::Windows::Forms::PictureBox^    pictureBox1;

private: System::Windows::Forms::PictureBox^    pictureBox2;

private: System::Windows::Forms::PictureBox^    pictureBox4;

protected:

protected:

protected:

protected:

protected:

private:
    /// <summary>
    /// Required designer variable.
    /// </summary>
    System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
    /// <summary>
    /// Required method for Designer support – do not modify
    /// the contents of this method with the code editor.
    /// </summary>
    void InitializeComponent(void)
    {
        System::ComponentModel::ComponentResourceManager^

```

```

resources = (gcnew System::ComponentModel::ComponentResourceManager(Form1::typeid));
    this->textBox1 = (gcnew System::Windows::Forms::TextBox());
    this->button1 = (gcnew System::Windows::Forms::Button());
    this->textBox3 = (gcnew System::Windows::Forms::TextBox());
    this->label4 = (gcnew System::Windows::Forms::Label());
    this->button2 = (gcnew System::Windows::Forms::Button());
    this->button4 = (gcnew System::Windows::Forms::Button());
    this->textBox2 = (gcnew System::Windows::Forms::TextBox());
    this->button5 = (gcnew System::Windows::Forms::Button());
    this->pictureBox1 = (gcnew System::Windows::Forms::PictureBox());
    this->pictureBox2 = (gcnew System::Windows::Forms::PictureBox());
    this->pictureBox4 = (gcnew System::Windows::Forms::PictureBox());
    (cli::safe_cast<System::ComponentModel::ISupportInitialize^
>(this->pictureBox1))->BeginInit();
    (cli::safe_cast<System::ComponentModel::ISupportInitialize^
>(this->pictureBox2))->BeginInit();
    (cli::safe_cast<System::ComponentModel::ISupportInitialize^
>(this->pictureBox4))->BeginInit();
    this->SuspendLayout();
    //
    //  textBox1
    //
    this->textBox1->Font = (gcnew System::Drawing::Font(L"Palatino L
        static_cast<System::Byte>(0)));
    this->textBox1->Location = System::Drawing::Point(314, 117);
    this->textBox1->Name = L"textBox1";
    this->textBox1->Size = System::Drawing::Size(138, 33);
    this->textBox1->TabIndex = 1;
    this->textBox1->TextChanged += gcnew System::EventHandler(this, &
    //
    //  button1
    //
    this->button1->BackColor = System::Drawing::Color::FromArgb(stat
        static_cast<System::Int32>(static_cast<System::Byte>(0))

```

```

this->button1->Font = (gnew System::Drawing::Font(L"Palatino Li
        static_cast<System::Byte>(0)));
this->button1->ForeColor = System::Drawing::SystemColors::Button
this->button1->Location = System::Drawing::Point(300, 171);
this->button1->Name = L"button1";
this->button1->Size = System::Drawing::Size(166, 91);
this->button1->TabIndex = 2;
this->button1->Text = L"Start Speaking";
this->button1->UseVisualStyleBackColor = false;
this->button1->Click += gnew System::EventHandler(this, &Form1:
//
// textBox3
//
this->textBox3->Font = (gnew System::Drawing::Font(L"Palatino L
        static_cast<System::Byte>(0)));
this->textBox3->Location = System::Drawing::Point(332, 301);
this->textBox3->Name = L"textBox3";
this->textBox3->Size = System::Drawing::Size(105, 33);
this->textBox3->TabIndex = 5;
//
// label4
//
this->label4->AutoSize = true;
this->label4->BackColor = System::Drawing::Color::Transparent;
this->label4->ForeColor = System::Drawing::Color::Red;
this->label4->Location = System::Drawing::Point(297, 155);
this->label4->Name = L"label4";
this->label4->Size = System::Drawing::Size(170, 13);
this->label4->TabIndex = 7;
this->label4->Text = L"Invalid Word!! Not in our dictionary";
this->label4->Visible = false;
//
// button2
//

```

```

this->button2->BackColor = System::Drawing::Color::Maroon;
this->button2->Font = (gcnew System::Drawing::Font(L"Palatino Li
        static_cast<System::Byte>(0)));
this->button2->ForeColor = System::Drawing::SystemColors::Button
this->button2->Location = System::Drawing::Point(458, 117);
this->button2->Name = L"button2";
this->button2->Size = System::Drawing::Size(47, 33);
this->button2->TabIndex = 9;
this->button2->Text = L"Clear\r\n";
this->button2->UseVisualStyleBackColor = false;
this->button2->Click += gcnew System::EventHandler(this, &Form1:
//
// button4
//
this->button4->BackColor = System::Drawing::Color::FromArgb(stat
        static_cast<System::Int32>(static_cast<System::Byte>(0)));
this->button4->Font = (gcnew System::Drawing::Font(L"Palatino Li
        static_cast<System::Byte>(0)));
this->button4->ForeColor = System::Drawing::SystemColors::Button
this->button4->Location = System::Drawing::Point(301, 171);
this->button4->Name = L"button4";
this->button4->Size = System::Drawing::Size(166, 91);
this->button4->TabIndex = 12;
this->button4->Text = L"Start Training";
this->button4->UseVisualStyleBackColor = false;
this->button4->Visible = false;
this->button4->Click += gcnew System::EventHandler(this, &Form1:
//
// textBox2
//
this->textBox2->Font = (gcnew System::Drawing::Font(L"Palatino L
        static_cast<System::Byte>(0)));
this->textBox2->Location = System::Drawing::Point(314, 117);
this->textBox2->Name = L"textBox2";

```

```

this->textBox2->Size = System::Drawing::Size(138, 33);
this->textBox2->TabIndex = 13;
this->textBox2->Visible = false;
//
// button5
//
this->button5->BackColor = System::Drawing::Color::Maroon;
this->button5->Font = (gcnew System::Drawing::Font(L"Palatino Li
        static_cast<System::Byte>(0)));
this->button5->ForeColor = System::Drawing::SystemColors::Button
this->button5->Location = System::Drawing::Point(458, 117);
this->button5->Name = L"button5";
this->button5->Size = System::Drawing::Size(47, 33);
this->button5->TabIndex = 14;
this->button5->Text = L"Clear";
this->button5->UseVisualStyleBackColor = false;
this->button5->Visible = false;
//
// pictureBox1
//
this->pictureBox1->BackColor = System::Drawing::Color::White;
this->pictureBox1->ErrorImage = (cli::safe_cast<System::Drawing:
>(resources->GetObject(L"pictureBox1.ErrorImage")));
this->pictureBox1->Image = (cli::safe_cast<System::Drawing::Image
>(resources->GetObject(L"pictureBox1.Image")));
this->pictureBox1->InitialImage = (cli::safe_cast<System::Drawin
>(resources->GetObject(L"pictureBox1.InitialImage")));
this->pictureBox1->Location = System::Drawing::Point(301, 260);
this->pictureBox1->Name = L"pictureBox1";
this->pictureBox1->Size = System::Drawing::Size(165, 74);
this->pictureBox1->TabIndex = 15;
this->pictureBox1->TabStop = false;
this->pictureBox1->Visible = false;
this->pictureBox1->WaitOnLoad = true;

```

```

        this->pictureBox1->Click += gnew System::EventHandler(this, &Form1::pictureBox1_Click);
//
// pictureBox2
//
this->pictureBox2->BackColor = System::Drawing::Color::White;
this->pictureBox2->Image = (cli::safe_cast<System::Drawing::Image>
>(resources->GetObject(L"pictureBox2.Image")));
this->pictureBox2->Location = System::Drawing::Point(546, 366);
this->pictureBox2->Name = L"pictureBox2";
this->pictureBox2->Size = System::Drawing::Size(112, 59);
this->pictureBox2->TabIndex = 16;
this->pictureBox2->TabStop = false;
this->pictureBox2->Click += gnew System::EventHandler(this, &Form1::pictureBox2_Click);
//
// pictureBox4
//
this->pictureBox4->BackColor = System::Drawing::Color::White;
this->pictureBox4->Image = (cli::safe_cast<System::Drawing::Image>
>(resources->GetObject(L"pictureBox4.Image")));
this->pictureBox4->Location = System::Drawing::Point(133, 366);
this->pictureBox4->Name = L"pictureBox4";
this->pictureBox4->Size = System::Drawing::Size(93, 59);
this->pictureBox4->TabIndex = 18;
this->pictureBox4->TabStop = false;
this->pictureBox4->Visible = false;
this->pictureBox4->Click += gnew System::EventHandler(this, &Form1::pictureBox4_Click);
//
// Form1
//
this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
this->BackColor = System::Drawing::SystemColors::ButtonFace;
this->BackgroundImage = (cli::safe_cast<System::Drawing::Image>
>(resources->GetObject(L"$this.BackgroundImage")));

```

```

        this->ClientSize = System::Drawing::Size(670, 449);
        this->Controls->Add(this->pictureBox4);
        this->Controls->Add(this->pictureBox2);
        this->Controls->Add(this->pictureBox1);
        this->Controls->Add(this->button5);
        this->Controls->Add(this->textBox2);
        this->Controls->Add(this->button4);
        this->Controls->Add(this->button2);
        this->Controls->Add(this->label4);
        this->Controls->Add(this->textBox3);
        this->Controls->Add(this->button1);
        this->Controls->Add(this->textBox1);
        this->ForeColor = System::Drawing::SystemColors::ActiveCaptionText;
        this->Name = L"Form1";
        this->Text = L"Speech Therapy";
        this->Load += gcnew System::EventHandler(this, &Form1::Form1_Load)
            (cli::safe_cast<System::ComponentModel::ISupportInitialize^>
>(this->pictureBox1))->EndInit();
            (cli::safe_cast<System::ComponentModel::ISupportInitialize^>
>(this->pictureBox2))->EndInit();
            (cli::safe_cast<System::ComponentModel::ISupportInitialize^>
>(this->pictureBox4))->EndInit();
        this->ResumeLayout(false);
        this->PerformLayout();

    }

#pragma endregion

private: System::Void button1_Click(System::Object^ sender, System::EventArgs^
e) {

        printf("\nhello!!");

    }

private: System::Void textBox1_TextChanged(System::Object^ sender, System::Even
e) {

```

```

        }
    private: System::Void label1_Click(System::Object^ sender, System::EventArgs^
e) {
        }
    private: System::Void button1_Click_1(System::Object^ sender, System::EventArgs^
e) {

        this->label4->Visible = false;

        String ^inputWord = textBox1->Text;
        char listenPath[100];
        sprintf(listenPath,"%s",inputWord);

        int k = checkWord(listenPath);

        if(checkWord(listenPath)){

            int speech_score = lTest(listenPath);

            this->textBox3->Text = speech_score.ToString();

        }else{

            this->label4->Visible = true;
        }

    }

    private: System::Void Form1_Load(System::Object^ sender, System::EventArgs^
e) {
        }
    private: System::Void textBox1_TextChanged_1(System::Object^ sender, System::EventArgs^
e) {

```



```

    }
private: System::Void button2_Click(System:: Object ^ sender , System::EventArgs ^
e) {

    this->textBox1->Text="";

}
private: System::Void linkLabel1_LinkClicked(System:: Object ^ sender , System::Windows::F
e) {

}
private: System::Void button3_Click(System:: Object ^ sender , System::EventArgs ^
e) {

    this->textBox3->Visible = false;
    this->button2->Visible = false;
    this->button1->Visible = false;
    this->textBox1->Visible = false;
    this->pictureBox1->Visible = true;
    this->textBox2->Visible = true;
    this->button4->Visible = true;
    this->button5->Visible = true;

}
private: System::Void button4_Click(System:: Object ^ sender , System::EventArgs ^
e) {

    String ^trainingWord = textBox2->Text;
    char listenPath[100];
    sprintf(listenPath,"%s" ,trainingWord);

    if (!checkWord(listenPath)){
        addNewWord(listenPath , ++word_number);
    }else{
        this->label4->Text = "The word is already pres
        this->label4->Visible = true;
    }

}

// addNewWord(trainingWord , ++word_number);

```

```

    }
private: System::Void pictureBox1_Click(System::Object^ sender, System::EventArgs^
e) {

    }
private: System::Void pictureBox2_Click(System::Object^ sender, System::EventArgs^
e) {

        this->textBox3->Visible = false;
        this->button2->Visible = false;
        this->button1->Visible = false;
        this->textBox1->Visible = false;
        this->pictureBox1->Visible = true;
        this->textBox2->Visible = true;
        this->button4->Visible = true;
        this->button5->Visible = true;
        this->pictureBox2->Visible = false;
        this->pictureBox4->Visible = true;

    }
private: System::Void pictureBox4_Click(System::Object^ sender, System::EventArgs^
e) {

        this->textBox3->Visible = true;
        this->button2->Visible = true;
        this->button1->Visible = true;
        this->textBox1->Visible = true;
        this->pictureBox1->Visible = false;
        this->textBox2->Visible = false;
        this->button4->Visible = false;
        this->button5->Visible = false;
        this->pictureBox4->Visible = false;
        this->pictureBox2->Visible = true;

    }

};
}

```