Indian Institute of Technology, Guwahati

Department of Computer Science and Engineering

Project Report on

Small Tutorial for Kids

Based on Speech Recognition System

Submitted to:

Prof. P. K. Das

Submitted by:

Prateekshya Priyadarshini (214101037)

Rohan Jaiswal (214101042)

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. Prateekshya Priyadarshini (214101037) 2. Rohan Jaiswal (214101042)

Contents

List of Figures

1 Abstract

This project is developed using C++/C. It can take a speech sample of a few seconds, preferably a single word, and display it's corresponding webpage which gives related information to the word. 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

This project uses a similar technique. There is a set of predefined words - aeroplane, ambulance, apple, autorickshaw, bicycle, bike, bus, car, cat, dog, scooter, tandem, train, tram, truck, orange and taxi. We can run the project and speak any of these words to see the related webpage. We can also train these words for new speakers. We can add new words which might take several minutes.

2.3 Future improvements

Since this project is developed using C/C++, it is difficult to run multiple things parallelly. Future improvements include development of this project using High Level Languages like Java or Python which can use multithreading concepts to run the model training part in background. This will remove the waiting time during training of new words.

3 Experimental Setup

Basic requirements for this project are as follows-

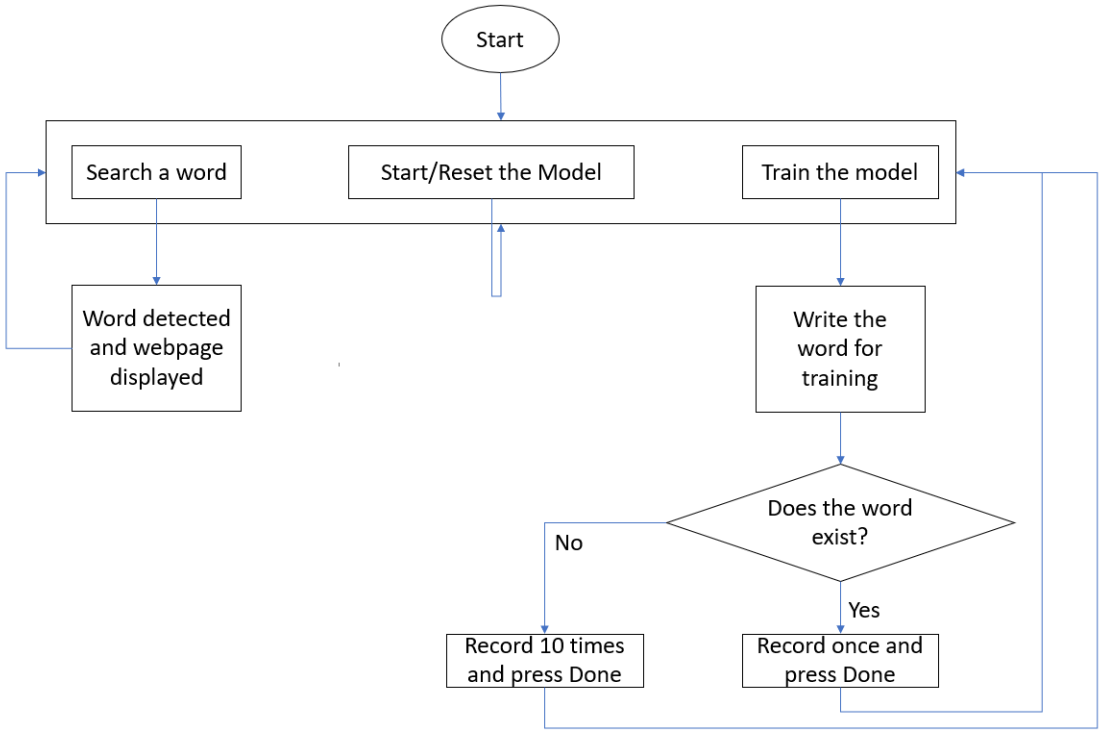

Figure 1: Flowchart of the project

- Windows OS
- 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 is 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 10 speech samples for each word.

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. If About button is clicked then Figure 3 will be displayed.

5.2 Live Testing

For live testing, corresponding button should be clicked. Then recording module will be opened which is showed in Figure 7. A word can be spoken and recorded for testing. The corresponding webpage will be displayed e.g. Figure 8.

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 i.e. Figure 5 or new i.e. Figure 6. If the word is already present, then recording will be allowed once, otherwise ten times. For recording, the same Figure 7 will be displayed. After recording, Done button can be clicked to get back to homepage i.e. Figure 2.

6 Source Code

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

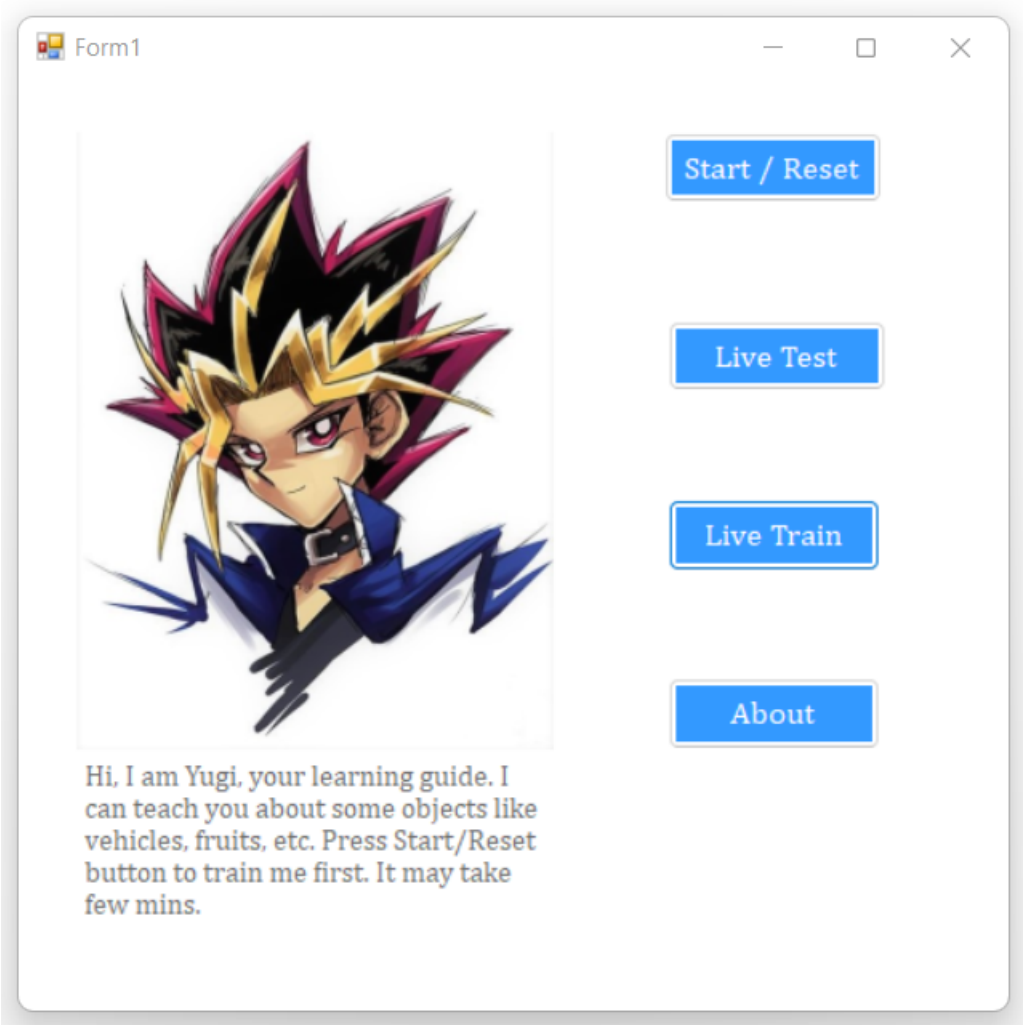

Figure 2: Home Page

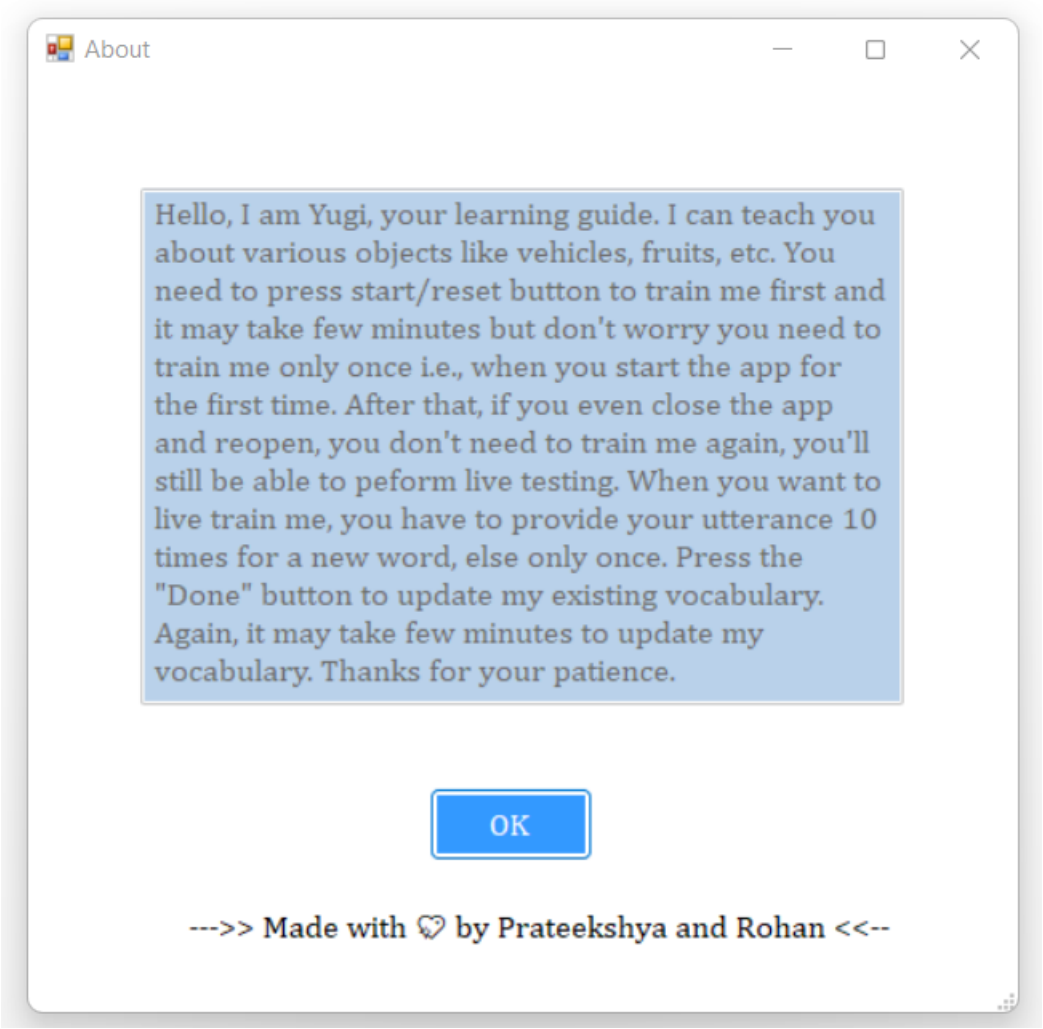

Figure 3: About Section

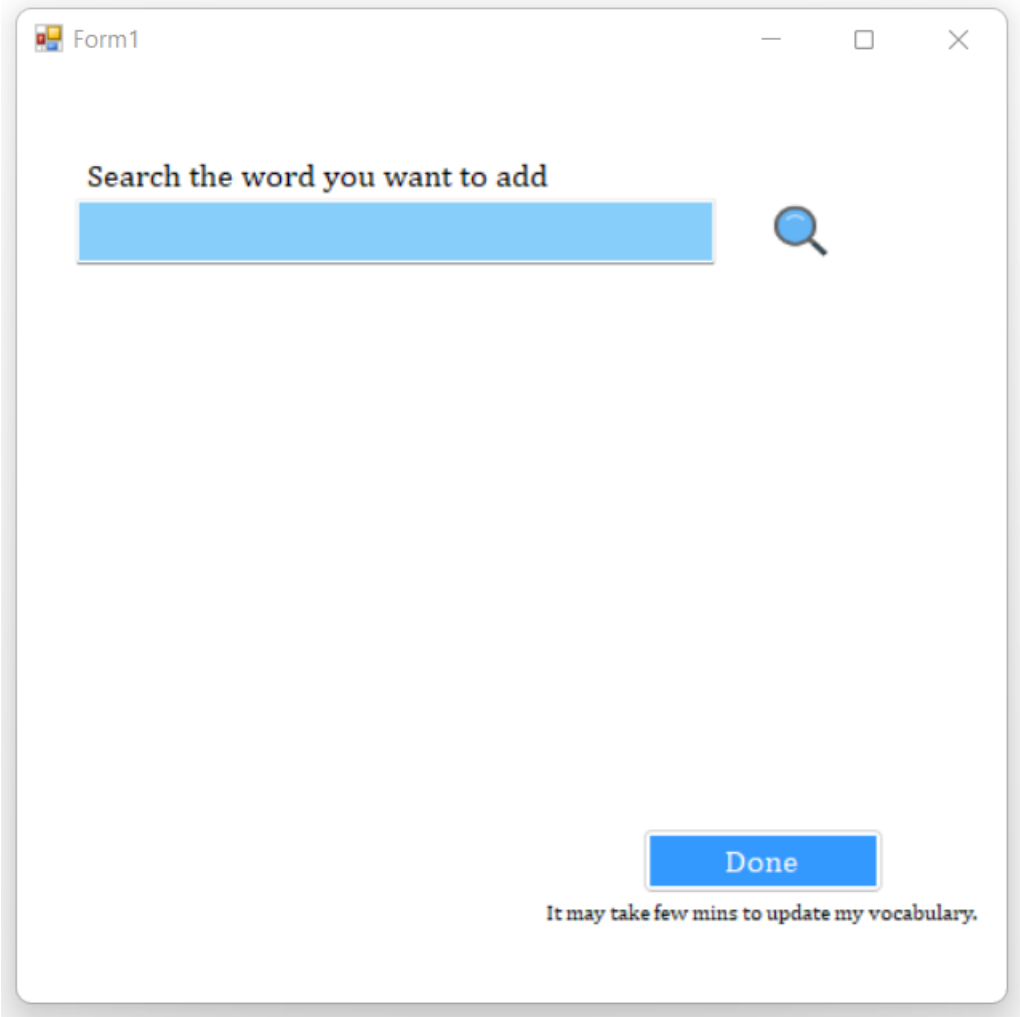

Figure 4: Live Training

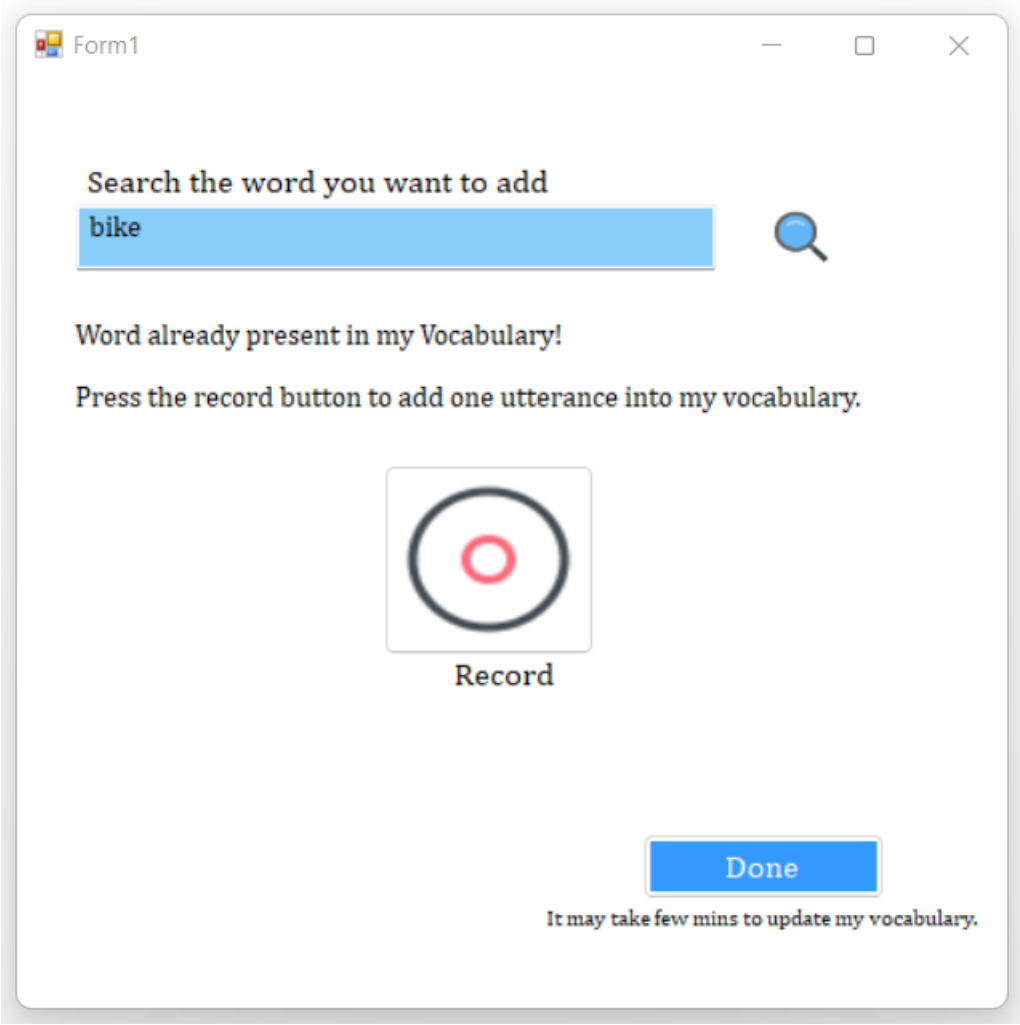

Figure 5: Existing Word Search

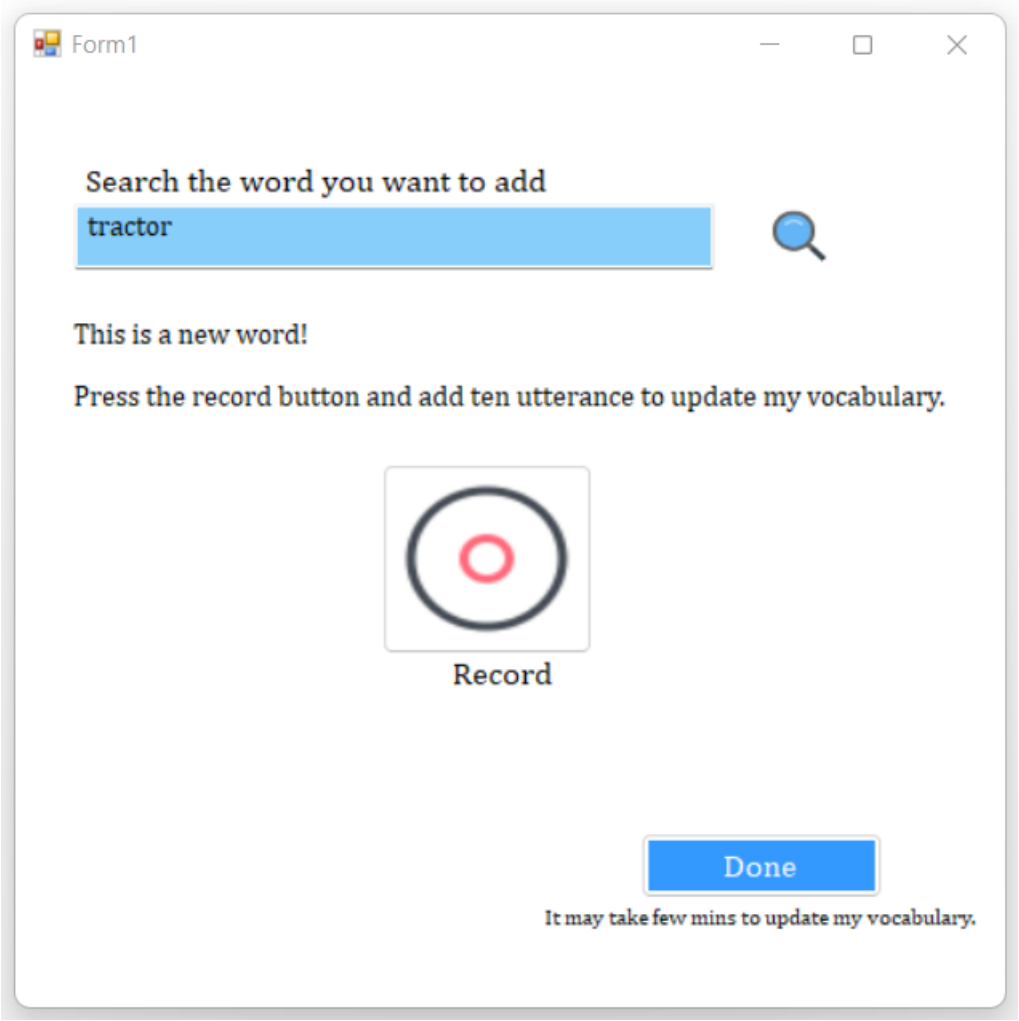

Figure 6: New Word Search

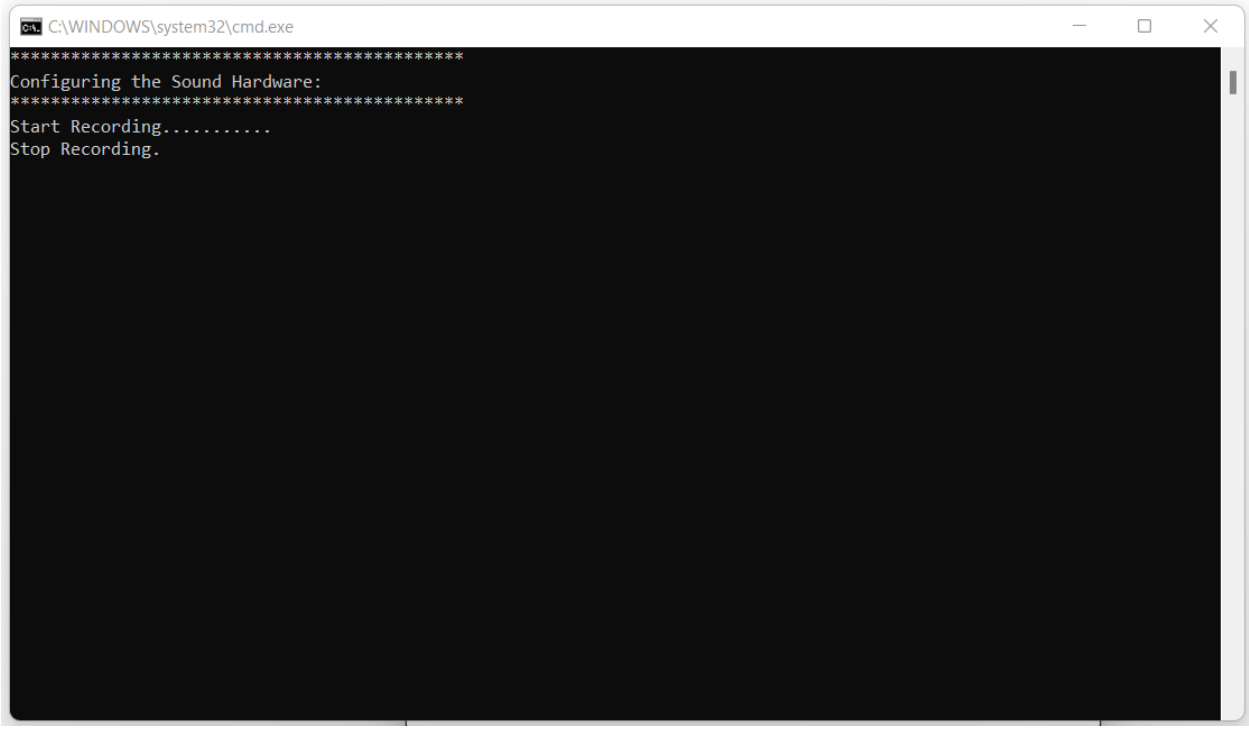

Figure 7: Recording Console

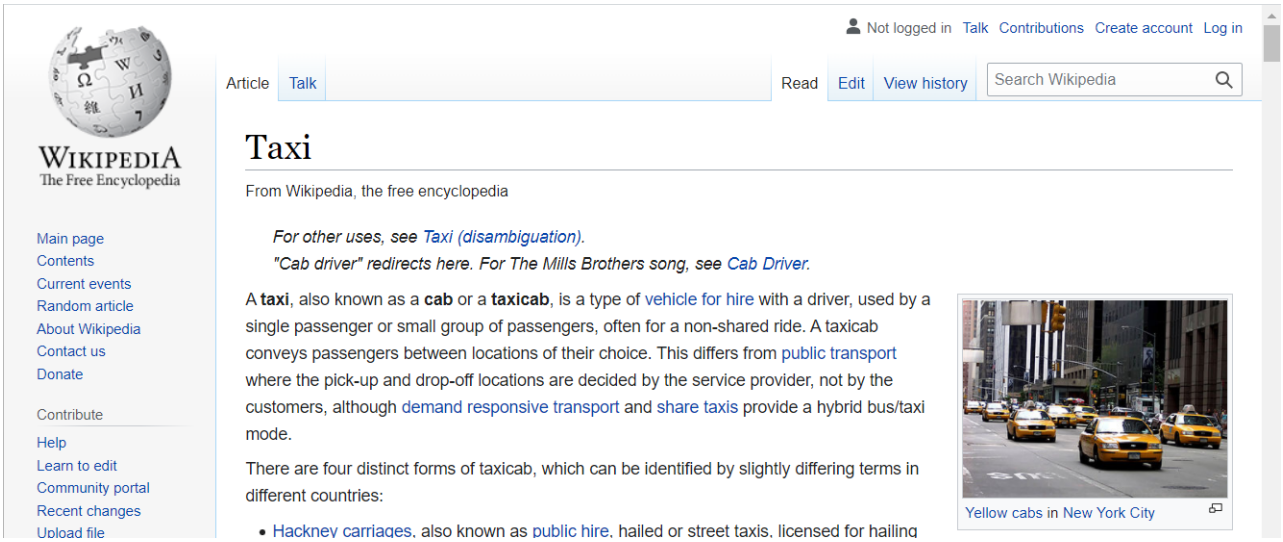

Figure 8: Display of Webpage

6.1 commonvar.h

```
long double *** xi = NULL; //xi in problem-3 solution


long double ** alpha = NULL; //Gets calculated in forward process

long double ** beta = NULL; //Gets calculated in backward process

long double ** delta = NULL; //Gets calculated in Viterbi Algorithm

long double ** gamma = NULL; //Gets calculated in Baum Welch method

long double ** A = NULL; //Transition matrix

long double ** AComplement = NULL; //updated A

long double ** B = NULL; //Probability matrix

long double ** BComplement = NULL; //updated B


long double * Pi = NULL; //Initial probability

long double * PiComplement = NULL; //updated Pi

long double ** codebook = NULL; //codebook

long double pOfOGivenLambda = 0; //probability of an observation sequence given the mode

long double pStar = 0; //probability of the state sequence being helpful in modelling

long double pStarComplement = 0; //probability of the state sequence being helpful in mo
```

```
long double floorB = 1e-30;


int ** psi = NULL; //Gets generated in Viterbi.h


int * O = NULL; //Observation sequences

int * qStar = NULL; //State sequence

int * qStarComplement = NULL; //State sequence for the updated model

char * resultWord = NULL;

int N = 0; //number of states

int M = 0; //codebook size or number of observations

int T = 100; //size of observation sequence

int R = 0; //read it from file, it will store the number of utterances per word currentl

int duration = 0; //read it from file, it will store the duration of the recording to be

int p = 12; //size of Codebook vectors

int universeSize = 0; //size of the universe

FILE * AComplementFile = NULL; //new A will be printed

FILE * BComplementFile = NULL; //new B will be printed

FILE * PiComplementFile = NULL; //new Pi will be printed
```

```
void define()
{
        int i = 0, j = 0;
        delta = new long double *[N];
        psi = new int *[N];
        qStar = new int[T];
        qStarComplement = new int[T];

        for (i = 0; i < N; ++i)
                delta[i] = new long double[T];
        for (i = 0; i < N; ++i)
                psi[i] = new int[T];

        xi = new long double ** [N];
        for (i = 0; i < N; ++i)
                xi[i] = new long double * [N];
        for (i = 0; i < N; ++i)
                for (j = 0; j < N; ++j)
                        xi[i][j] = new long double[T-1];

        gamma = new long double * [N];
        for (i = 0; i < N; ++i)
                gamma[i] = new long double[T];

        PiComplement = new long double [N];

        AComplement = new long double * [N];
        for (i = 0; i < N; ++i)
                AComplement[i] = new long double[N];

        BComplement = new long double * [N];
        for (i = 0; i < N; ++i)
                BComplement[i] = new long double[M];
```

```
        O = new int [N]; //Observation Sequence
}

FILE * dataOutputFile = NULL; //output file for required output

FILE * modelOutputFile = NULL; //output file for model
```

6.2 About.h

```
#pragma once

namespace Yugi {

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

        protected:
```

```
		/// <summary>
		/// Clean up any resources being used.
		/// </summary>
		~About()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::Button^  okBtn;
	protected:
	private: System::Windows::Forms::TextBox^  aboutText;

	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(About::typeid));
			this->okBtn = (gcnew System::Windows::Forms::Button());
			this->aboutText = (gcnew System::Windows::Forms::TextBox());
			this->SuspendLayout();
			//
			// okBtn
```

```
                //
                this->okBtn->Location = System::Drawing::Point(99, 216);
                this->okBtn->Name = L"okBtn";
                this->okBtn->Size = System::Drawing::Size(110, 45);
                this->okBtn->TabIndex = 0;
                this->okBtn->Text = L"OK";
                this->okBtn->UseVisualStyleBackColor = true;
                this->okBtn->Click += gcnew System::EventHandler(this, &About::o
                //
                // aboutText
                //
                this->aboutText->Enabled = false;
                this->aboutText->Location = System::Drawing::Point(28, 33);
                this->aboutText->Multiline = true;
                this->aboutText->Name = L"aboutText";
                this->aboutText->Size = System::Drawing::Size(274, 166);
                this->aboutText->TabIndex = 1;
                this->aboutText->Text = resources->GetString(L"aboutText.Text");
                //
                // About
                //
                this->AutoScaleDimensions = System::Drawing::SizeF(8, 16);
                this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Fon
                this->ClientSize = System::Drawing::Size(330, 298);
                this->Controls->Add(this->aboutText);
                this->Controls->Add(this->okBtn);
                this->Name = L"About";
                this->Text = L"About";
                this->ResumeLayout(false);
                this->PerformLayout();

            }
#pragma endregion
        private: System::Void okBtn_Click(System::Object^ sender, System::EventArgs^
```

```
e) {
                                        Form::Close();
                                }
        };
}
```

6.3 Form1.h

```
#pragma once
// #include "LiveTrain.h"
#include <msclr\marshal_cppstd.h>
#include "About.h"
#include "FinalProject.h"

namespace Yugi {

        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::Button^  liveTrainBtn;
private: System::Windows::Forms::Button^  liveTestBtn;
private: System::Windows::Forms::Button^  resetBtn;

private: System::Windows::Forms::Button^  aboutBtn;
private: System::Windows::Forms::TextBox^  searchTextBox;
private: System::Windows::Forms::Button^  searchBtn;
private: System::Windows::Forms::Label^  searchLabel;
private: System::Windows::Forms::Button^  recordBtn;
private: System::Windows::Forms::Button^  doneBtn;


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)
            {
                this->liveTrainBtn = (gcnew System::Windows::Forms::Button());
                this->liveTestBtn = (gcnew System::Windows::Forms::Button());
                this->resetBtn = (gcnew System::Windows::Forms::Button());
                this->aboutBtn = (gcnew System::Windows::Forms::Button());
                this->searchTextBox = (gcnew System::Windows::Forms::TextBox());
                this->searchBtn = (gcnew System::Windows::Forms::Button());
                this->searchLabel = (gcnew System::Windows::Forms::Label());
                this->recordBtn = (gcnew System::Windows::Forms::Button());
                this->doneBtn = (gcnew System::Windows::Forms::Button());
                this->SuspendLayout();
                //
                // liveTrainBtn
                //
                this->liveTrainBtn->Location = System::Drawing::Point(241, 238);
                this->liveTrainBtn->Name = L"liveTrainBtn";
                this->liveTrainBtn->Size = System::Drawing::Size(142, 44);
                this->liveTrainBtn->TabIndex = 2;
                this->liveTrainBtn->Text = L"Live Train";
                this->liveTrainBtn->UseVisualStyleBackColor = true;
                this->liveTrainBtn->Click += gcnew System::EventHandler(this, &F
                //
                // liveTestBtn
                //
                this->liveTestBtn->Location = System::Drawing::Point(60, 238);
                this->liveTestBtn->Name = L"liveTestBtn";
                this->liveTestBtn->Size = System::Drawing::Size(145, 43);
```

```
this->liveTestBtn->TabIndex = 3;
this->liveTestBtn->Text = L"Live Test";
this->liveTestBtn->UseVisualStyleBackColor = true;
this->liveTestBtn->Click += gcnew System::EventHandler(this, &Fo
//
// resetBtn
//
this->resetBtn->Location = System::Drawing::Point(60, 303);
this->resetBtn->Name = L"resetBtn";
this->resetBtn->Size = System::Drawing::Size(145, 43);
this->resetBtn->TabIndex = 5;
this->resetBtn->Text = L"Start/Reset";
this->resetBtn->UseVisualStyleBackColor = true;
this->resetBtn->Click += gcnew System::EventHandler(this, &Form1
//
// aboutBtn
//
this->aboutBtn->Location = System::Drawing::Point(241, 303);
this->aboutBtn->Name = L"aboutBtn";
this->aboutBtn->Size = System::Drawing::Size(142, 44);
this->aboutBtn->TabIndex = 4;
this->aboutBtn->Text = L"About";
this->aboutBtn->UseVisualStyleBackColor = true;
this->aboutBtn->Click += gcnew System::EventHandler(this, &Form1
//
// searchTextBox
//
this->searchTextBox->Location = System::Drawing::Point(30, 31);
this->searchTextBox->Name = L"searchTextBox";
this->searchTextBox->Size = System::Drawing::Size(256, 22);
this->searchTextBox->TabIndex = 6;
//
// searchBtn
//
```

```
this->searchBtn->Location = System::Drawing::Point(312, 31);
this->searchBtn->Name = L"searchBtn";
this->searchBtn->Size = System::Drawing::Size(87, 21);
this->searchBtn->TabIndex = 7;
this->searchBtn->Text = L"Search";
this->searchBtn->UseVisualStyleBackColor = true;
this->searchBtn->Click += gcnew System::EventHandler(this, &Form
//
// searchLabel
//
this->searchLabel->AutoSize = true;
this->searchLabel->Location = System::Drawing::Point(37, 81);
this->searchLabel->Name = L"searchLabel";
this->searchLabel->Size = System::Drawing::Size(85, 17);
this->searchLabel->TabIndex = 8;
this->searchLabel->Text = L"search label";
//
// recordBtn
//
this->recordBtn->Location = System::Drawing::Point(146, 151);
this->recordBtn->Name = L"recordBtn";
this->recordBtn->Size = System::Drawing::Size(140, 39);
this->recordBtn->TabIndex = 9;
this->recordBtn->Text = L"Record";
this->recordBtn->UseVisualStyleBackColor = true;
this->recordBtn->Click += gcnew System::EventHandler(this, &Form
//
// doneBtn
//
this->doneBtn->Location = System::Drawing::Point(241, 288);
this->doneBtn->Name = L"doneBtn";
this->doneBtn->Size = System::Drawing::Size(142, 41);
this->doneBtn->TabIndex = 10;
this->doneBtn->Text = L"Done";
```

```
                this->doneBtn->UseVisualStyleBackColor = true;
                this->doneBtn->Click += gcnew System::EventHandler(this, &Form1:
                //
                // Form1
                //
                this->AutoScaleDimensions = System::Drawing::SizeF(8, 16);
                this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Fon
                this->ClientSize = System::Drawing::Size(437, 397);
                this->Controls->Add(this->doneBtn);
                this->Controls->Add(this->recordBtn);
                this->Controls->Add(this->searchLabel);
                this->Controls->Add(this->searchBtn);
                this->Controls->Add(this->searchTextBox);
                this->Controls->Add(this->resetBtn);
                this->Controls->Add(this->aboutBtn);
                this->Controls->Add(this->liveTestBtn);
                this->Controls->Add(this->liveTrainBtn);
                this->Name = L"Form1";
                this->Text = L"Form1";
                this->Load += gcnew System::EventHandler(this, &Form1::Form1_Loa
                this->ResumeLayout(false);
                this->PerformLayout();

            }
#pragma endregion
        private: System::Void Form1_Load(System::Object^  sender, System::EventArgs^
e) { //all the components required for live training are disabled
                    searchBtn->Visible = false;
                    searchBtn->Enabled = false;
                    searchTextBox->Visible = false;
                    searchTextBox->Enabled = false;
                    searchLabel->Visible = false;
                    searchLabel->Enabled = false;
                    recordBtn->Visible = false;
```

```
                                recordBtn->Enabled = false;
                                doneBtn->Visible = false;
                                doneBtn->Enabled = false;
                        }
        //private: System::Void startBtn_Click(System::Object^  sender, System::EventArg
e) {
        //                       startBtn->Visible = false;
        //                       liveTestBtn->Visible = true;
        //                       liveTrainBtn->Visible = true;
        //                       aboutBtn->Visible = true;
        //                       resetBtn->Visible = true;
        //                }

        private: System::Void liveTrainBtn_Click(System::Object^  sender, System::EventA
e) { //disable the previous components and enable the components required for live traini
                                /*LiveTrain^ liveTrain = gcnew LiveTrain;
                                liveTrain->ShowDialog();*/
                                aboutBtn->Visible = false;
                                aboutBtn->Enabled = false;
                                resetBtn->Visible = false;
                                resetBtn->Enabled = false;
                                liveTestBtn->Visible = false;
                                liveTestBtn->Enabled = false;
                                liveTrainBtn->Visible = false;
                                liveTrainBtn->Enabled = false;
                                searchBtn->Visible = true;
                                searchBtn->Enabled = true ;
                                searchTextBox->Visible = true;
                                searchTextBox->Enabled = true;
                                searchLabel->Visible = true;
                                searchLabel->Enabled = true;
                                recordBtn->Visible = true;
                                recordBtn->Enabled = true;
                                doneBtn->Visible = true;
```

```
                                doneBtn->Enabled = true;
                            }
        private: System::Void liveTestBtn_Click(System::Object^  sender, System::EventAr
e) { //browse the wikipedia link for the detected word
                            // System::String^ str = gcnew String(res);
                            char *str1 = "start https://en.wikipedia.org/wiki/";
                            char *str2 = performLiveTesting();
                            char *str3 = (char *)malloc(1 + strlen(str1)+ strlen(str2));
                            strcpy(str3, str1);
                            strcat(str3, str2);
                            // System::String^ str = gcnew String(str2);
                            // Greeting->Text = str;
                            system(str3);
                    }
private: System::Void aboutBtn_Click(System::Object^  sender, System::EventArgs^
e) { //for about section
                            About^ about = gcnew About;
                            about->ShowDialog();
                    }
private: System::Void resetBtn_Click(System::Object^  sender, System::EventArgs^
e) {
                                start(); //initial model training
                    }
private: System::Void searchBtn_Click(System::Object^  sender, System::EventArgs^
e) { //search the word and display the number of recordings required
                                String^ strr = searchTextBox->Text;
                                msclr::interop::marshal_context context;
                                std::string str= context.marshal_as<std::string>(strr);
                                int found = searchWord(str.c_str());
                                if(found == 1) {
                                        searchLabel->Text = "Record Once";
                                } else {
                                        searchLabel->Text = "Record Ten times";
                                }
```

```
                }
private: System::Void recordBtn_Click(System::Object^  sender, System::EventArgs^
e) {
                                recordWords();
                }
private: System::Void doneBtn_Click(System::Object^  sender, System::EventArgs^
e) { //switch the interface to the previous menu
                         performLiveTraining();
                         aboutBtn->Visible = true;
                        aboutBtn->Enabled = true;
                        resetBtn->Visible = true;
                        resetBtn->Enabled = true;
                        liveTestBtn->Visible = true;
                        liveTestBtn->Enabled = true;
                        liveTrainBtn->Visible = true;
                        liveTrainBtn->Enabled = true;
                                searchBtn->Visible = false;
                                searchBtn->Enabled = false ;
                                searchTextBox->Visible = false;
                                searchTextBox->Enabled = false;
                                searchLabel->Visible = false;
                                searchLabel->Enabled = false;
                                recordBtn->Visible = false;
                                recordBtn->Enabled = false;
                                doneBtn->Visible = false;
                                doneBtn->Enabled = false;
                }
};
}
```