Information searching through voice to text conversion at call centers

Samaranayaka J. R. A. C. P.
149230M

Faculty of information technology
University of moratuwa
May 2017

Information searching through voice to text conversion at call centers

Samaranayaka J. R. A. C. P.
149230M

Dissertation submitted to the Faculty of Information Technology, University of Moratuwa, Sri Lanka for the partial fulfillment of the requirements of the Master Degree of Science in Information Technology.
May 2017

Declaration

I declare that this thesis is my own work and has not been submitted in any form for another degree or diploma at any university or other institution of tertiary education. Information derived from the published or unpublished work of others has been acknowledged in the text and a list of references is given.

Name of the student: Samaranayaka J. R. A. C. P.

Student Number: 149230M

Signature of the student: ……..... Date: …….....

Supervised by

Name of the supervisor: Mr. Saminda Premaratne

Signature of the supervisor: ……..... Date: ……

Dedication

I would like to dedicate my dissertation to my parents for encouraging me to follow Master Degree of Science in Information Technology.

Acknowledgement

Firstly, I want to thank my supervisor Mr.Saminda Premaratne for the guidance given to make success the research in time. Next, I want to thank Prof. Asoka S Karunanda for conducting Research Methodology and Literature Review and Thesis Writing subjects which are helped me a lot to successfully complete the research. Then, I want to thank the lecture panel of the Faculty of Information Technology, University of Moratuwa. Finally, I want to thank each and everyone who helped me to make success my MSc research project.

Abstract

A call center is a central place where Customer Care people handle queries of customers over the telephone to fulfill their needs. Information searching delay is an increasingly problem in a call center because of multiple subsystems in operation. Delay of a one customer may lead to dissatisfaction of the others due to queue for services. This will affect to the call centers key performance indicators negatively. We have conducted our research to reduce the thinking time and key board entering time by using voice to text as the main technology. This research uses several technologies namely capturing users' key strokes, information categorization and focusing web browser's information as a response to a keystroke other than voice to text. Our applications prime goal is to reduce the queues for services and increase the customer satisfaction by reducing the service delay by using voice to text technology.

Our application is a standalone application which is not needed to install and uses .NET framework and NAudio library for enhancing and converting voice signal into text. Our application is consisted with a prediction algorithm which is capable for correcting falsely recognized text. This will increase the accuracy of the recognized sentences and this will lead to better information searching functionality. After the conversion, application will perform the searching and categorization by using naïve bayes algorithm and HTTP communication which will not lead to any security constraints in an organization because of no need any port opening requirement with the support of content management system which is managed centrally and loosely coupled with the desktop application.

Application has the capability of maintaining consistence searching and categorization functionality in a case of sound input failure by taking user's key inputs instead of sound inputs. Users of the application no need to follow highly time and human memory consuming file menus or hyperlinks. Categorization algorithm will proposed the most probable locations of the required information.

Application was tested with several users and the accuracy of the information categorization was measured by taking a sample content from the call centers technical FAQ site as the metadata. It was 90% accurately categorized the text in to

groups and subgroups. Although voice to text conversion accuracy is varying with the sound cards performance, our prediction algorithm will increase the accuracy of the voice to text conversion than the normal voice to text conversion mechanisms. Finally, application was 100% successfully queried the information from content management system via HTTP communication and display it on the web browser.

Table of Contents

List of Figures

List of Tables

Chapter1

Introduction

1.1 Prolegomena

A call center is a central place where Customer Care people handle queries of customers over the telephone to fulfill their needs. In a call center several metrics will be used to measure the performance such as Service level, Quality monitoring scores, Customer satisfaction, Adherence, Occupancy, Average handle time, Number of calls offered, First call resolution rate, etc. Average handle time is the total average amount of time an agent spends talking and in post call work in relation to a call. This is a metric which is directly related to agent skill and his/her performance. Average handle time will increase due to low skill level of the agent, inability to clearly understand the customer's query, delay of accessing relevant information, not up-to-date details of the systems, less search ability of the systems, less user-friendliness of the systems, etc. In this research, we are focusing on reducing the delay of accessing information which leads to reducing Average handle time.

1.2 Background and motivation

Call centers will provide services to the customers over the telephone. Typically call centers will handle both inbound which will be generated from the outside and request customer support, help-desk services, reservation and sales support and outbound which will be generated by call center agents to the customers and conduct marketing campaigns, surveys, etc traffics (Gans et al., 2003). When the waiting time increases in the call center, it will lead to customer dissatisfaction (McGuire et al., 2010). Queue time and handling time is the total time customer spent to fulfill his/her requirement (Gans et al., 2003). Handling time will be increased due to barriers in information intelligence which is ability to successfully search, assemble all pieces, analyze, and effectively use all relevant available information within the organization (Evgeniou and Cartwright, 2005). Those barriers are categorized into three types namely Behavioral barriers, Process barriers and Organizational barriers.

Customer handling time can be reduced by training the call center agents and

employing proper tools and technologies to query customer requests efficiently (Rasooli and Albadvi, 2007). Major media in a call center is voice calls, although other access methods are available to customers such as web, text, sms, mail, etc (Kumar and Telang, 2012). We are using voice to text technology (Lee and Oun-Young, 1988; Tuerk et al., 1991) to reduce the typing delay and understanding time of the customer query by using sentiment analyzing (Medhat et al., 2014).

1.3 Problem in brief

Customers are preferred to contact call center agents over self-service, it is need to improve call handling time of the call center. Agents will access several subsystems to retrieve information to fulfill the customers' request. This will induce delay, because of searching inefficiency and bringing searched or opened systems to foreground on demand with less effort.

1.4 Hypothesis

We can improve efficiency of information searching at call centers by introducing voice to text technology for fast inputs to the systems.

1.5 Aim

- Reduce the queues for services in call centers and increase the customer satisfaction by reducing the service delay by using voice to text technology

1.6 Objectives

- Improve searching speed to fulfill customer satisfaction. We should minimize the searching delay to reduce the customer waiting time and call handling duration. This will lead to handling high call volume in peak time to improve service level.

- Analyze level of technical knowledge required against time. This will help to train staffs with most mandatory knowledge and improve staff scheduling to satisfy customers.

- Prove the proposed solution by using scientific method.

- Collect feedbacks/data from the users (Agents, etc.).

1.7 Proposed solution

This is a standalone application which has the capability to combine user's key input and voice to construct the key word array which is needed to search. At the input it has the capability of suggesting words when the user input's spellings are wrong. Next phase it will use text mining to enhance the key word combination. This will also lead to categorization of the voice clip (high tech, medium tech, low complexity, etc.). Then it will save key word combinations and category and time stamp for reporting purposes. Those data will be used for analyzing the demand.

As the next step we will search information by using key word combination which was constructed at the previous step and it will show on the web browser. Web searching function can be enabled by using dynamic URL construction and the CMS searching functionality.

Finally proposed solution has the functionality of focusing already displayed details on the browser by using key strokes to minimize manual selection delay by the user of the software.

1.8 Structure of thesis

The rest of the thesis is organized as follows. Chapter two critically reviews the research in voice to text for call centers and defines the research problem together with the identification of the technology to solve the problem. Chapter3 is a detailed description on voice to text and web technologies. Chapter4 is the appoarch of the voice to text for information searching. Chapter5 is the design and Chapter6 is the implementation of the proposed solution. Chapter7 will present the evaluation results of the voice to text approach for information searching. Chapter8 concludes the research finding with note on further work.

1.9 Summery

This chapter presented background and motivation, problem in brief, hypothesis, objectives of the research and proposed solution. As explained in this chapter, we are trying to reduce the information searching delay in call centers by using voice to text technology and sentiment analysis techniques. Next chapter is the literature review chapter and it will provide a detailed description about the back ground of the problem.

Chapter2

Litreture survey

2.1 Introduction

Chapter 1 gave a comprehensive description of the overall project. This chapter provides a critical review of the literature in related to developments and challenges in information searching in call centers. For this purpose, the review of the past researches has been presented under two major sections namely importance of the efficient information access and technologies in the call center. At the end this chapter defines the research problem as delay of searching information to increase the call handling time and identifies the voice technologies to be used to address the problem.

2.2 Importance of the efficient information access

Danilo Garcia and others have conducted a research on managing time and customer satisfaction at call centers(Garcia et al., 2012). This research based on the study of relationship between customer satisfaction and waiting time for the service. Organization with intelligent information regarding customers uses them as an asset to their success (Evgeniou and Cartwright, 2005). Waiting time is an important factor in the service industry because increased waiting time will lead to customer dissatisfaction (McGuire et al., 2010). This will lead to work strategy of the organization. This research suggests managers and decision makers might need to know which variables might influence recollections of satisfaction with waiting time in the queue. Also research presented that people are more willing to wait in line depending on how much they value the service provided by the organization. Limitation in the research is only few variables were used. This research concluded that quality service and excellent information is more valuable when customers report recollections of waiting time in the queue.

Mathew and others have done a study on business intelligence and its implications for call center(Dabrowski, 2013). This study was based on importance of business intelligence on making operational and organizational decisions (Khan and Quadri, 2012). Business intelligent is a product of information and knowledge (Dabrowski,

2013). Information is the meaningful interpretation of raw data which are consisted of collected facts, observation or perception (Sabherwal and Becerra-Fernandez, 2011) while knowledge is a conclusion or collection of conclusions driven by the information (Sabherwal and Becerra-Fernandez, 2011). Organizational memory, information integrity which is collected via various types of text and web mining, insight creation and presentation capabilities are the four key areas of technologies which are enabling the business intelligence (Dabrowski, 2013; Sabherwal and Becerra-Fernandez, 2011). Key performance indicators help to measure the progress in the call center towards the organizational goals (Garcia et al., 2012). As results shown, values of the key performance indicators will help to organize agent training programs, customer satisfactions improvement projects and technological improvements to decrease handle time (Dabrowski, 2013). Business intelligence will not completely eliminate the manual intervention in call center. As results shown, investigations were conducted over the Key performance indicators to find the actual problem manually (Dabrowski, 2013).

Pooya and colleagues have done a study on knowledge management in call centers (Rasooli and Albadvi, 2007). This study was based on managing different roles of knowledge in order to reach the optimum efficiency and competitive advantage through customer satisfaction (Rasooli and Albadvi, 2007). This research focuses on knowledge acquisition, utilization, adaption, dissemination and generation. Three different kinds of knowledge can be gained by call centers by interacting with customers (Tsoukas and Vladimirou, 2001). The process designed to deal with call center customers is CRM (Kotsovos and Kriemadis, n.d.; Mozaheb et al., 2015) which starts from building customer knowledge and maintain profitable customer relationships. Swift proposed a recursive model which consists with knowledge discovery, market planning, customer interaction, analysis and refinement for CRM (Rasooli and Albadvi, 2007). Oluic-Vukovic proposed a five step process which consists with gathering, organizing, refining, representing and disseminating for knowledge acquisition (Rasooli and Albadvi, 2007). Wingens divided knowledge utilization theories into three major categories as knowledge specific theories, policy maker constraints theories and two community theories (Rasooli and Albadvi, 2007). Others also proposed models for knowledge utilization (Becheikh et al., 2010; Rasooli and Albadvi, 2007). Although communication infrastructure can easily deliver

information, it can create flood of information for call center staff by resulting an overload.

2.3 Technologies in the call center

Devina and others have done a study on technology in a call center (Oodith and Parumasur, 2014). This study was based on influence of technology on call center agents' effectiveness in managing customers and their needs (Oodith and Parumasur, 2014). Understanding and ease use of technology for effectively managing customers and their needs differ from agent to agent varying in biographical profiles is the hypothesis for this research. Although internet based applications lead to competitiveness, efficiency and globalization, customers would prefer personal interactions rather than self-services in call centers (Oodith and Parumasur, 2014). Fitzsimmons and others identified five modes of technology's contribution to the service as technology free service, technology assisted service, technology facilitated service, technology mediated service and technology generated service (Fitzsimmons, 2005). It is a mistake to go with automated low cost self-services by ignoring the human side of the customer (Oodith and Parumasur, 2014). Unplanned or unsuitable technology enhancements will lead to additional stress on call center staff (Oodith and Parumasur, 2014). Research paper introduced blogs (Larsson and Hrastinski, 2011) which assist in sharing information between the firm and the customer by avoiding too many information emails to customers' spam folder as a new variation to chat room. According to the results of the research, Self service will be a benefit for both firm and the customer in many ways such as cost reduction, staff reduction, customized services, accuracy, convenience and speed. Most suitable ratio which should be maintained in a call center between the self-service and the call center agents is not properly discussed in this paper.

Anuj and others have done a study on "web reduces customer service cost" (Kumar and Telang, 2012). This study was based on web technology and its impact on call centers (Kumar and Telang, 2012). Call centers will be exposed to rich customer interaction data that can be analyzed to provide customized goods and services to customers (Kumar and Telang, 2012). Because direct labor cost is the major cost component (Lee et al., 2007) in a call center, firms are willing to implement self-

services for customers such as IVR and web based self-services. Role clarity, motivation and ability are key mediators which lead to increasingly use of self-services (L. Michelle Bobbitt and Pratibha A. Dabholkar, 2001; Meuter et al., 2005). According to the results, it's shown that uses who adopted for web based self-services which contains uncertain and complex information reduce the IVR usage. But, increase their consumption of the call center (Kumar and Telang, 2012). Research concluded that introducing one channel does not necessarily reduce the consumption in another channel. However research did not provide sufficient details about the impact to the call handling duration as a result of a web based self-service implementation.

Christine Tuerk and others have conducted a research on the development of a connectionist multiple voice text-to-speech (Tuerk et al., 1991). Others also conducted similar kind of researches in the area of text-to-speech (DobriSek et al., 1999; Lee and Oun-Young, 1988). This research is based on the theory letter to phoneme mapping and phonemes onto LPC synthesizer. This research paper focusing on generating speech synthesis rules automatically over traditional hand written rules. It was observed that a comparable performance improvement by employing above method by using listening tests. But this gave good results for single speaker speech synthesis only.

Mohan and others have conducted a survey on algorithms used for sentiment analysis (Medhat et al., 2014). Sentiment analyzing (SA) is that getting the idea of people about a specific product, movies, organization, news, events, services, issues, etc. SA identifies the sentiments embedded in the text and then analyses that. There are three main classification levels in SA such as Document Level which aims to classify an opinion document and express whether that is a positive or negative opinion or sentiment, Sentence Level which aims to classify emotion expressed in each sentence and Aspect Level which aims to classify with respect to the precise aspects of entities.SA techniques can be divided into three categories such as machine learning approach which is sub divided into supervised learning and unsupervised learning, lexicon based approach and hybrid approach. NaiveBayes classification model which is the most frequently used and simplest method computes the posterior probability of a class, based on the distribution of the words in the document. This method is fallen

under the supervised learning category and uses Bayes Theorem to predict the probability.

$P(h/D) = [P(D/h)\ P(h)]/P(D)$

Where,

P(h) : Prior probability of hypothesis h (predicted value)

P(D) : Prior probability of training data D (value given)

P(h/D) : Probability of h given D (expected o/p)

2.4 Problem definition

This literature review identified various unsolved problems including delay of searching information and less intelligent monitoring of agent's activities. Since customers are preferred to contact call center agents over self-service, it is need to improve call handling time of the call center. Agents will access several subsystems to retrieve information to fulfill the customers' request. Therefore we need to improve agent's efficiency of retrieving information than forcing customers for self-service to fulfill the requirements. We have noticed that limited researches in voice to text for information searching in call centers has been recorded in the literature.

2.5 Summery

This chapter presented a comprehensive literature review on the information which is used at call center and identified the research problem as the delay of searching information to increase the call handling time. We also identified the voice to text technology to address above problem. Next chapter will discuss the technology to be used for our solution.

Chapter 3

Theory of voice to text conversion

3.1 Introduction

In this chapter, we are describing the architecture, technologies and algorithms (theories) used to develop the application. As shown in figure 3.1, application is consisted with four layers such as communication layer, input enhancement layer, conversion layer, and business logic. Bellow sub topics will describe the main technologies and theories used in each layer separately. Communication layer is consisted with several sub modules such as HTTP handler, voice input handler, key stroke detector, tab handler and DB client. Input enhancement layer is consisted with URL constructer, voice input preprocessor, key stroke decoder, and tab command constructer. Conversion layer has two modules such as spelling corrector module and voice to text conversion module.

Proposed solution is a standalone application which is written in C# programming language and C++ programming language. Application operates on ".NET 4 framework" and Windows operating system. "Microsoft Visual C# 2010 Express" was used as the developing tool for the proposed application. Wamp sever and PHP programming language was used to test the HTTP communication in-between HTTP handler and the web server.

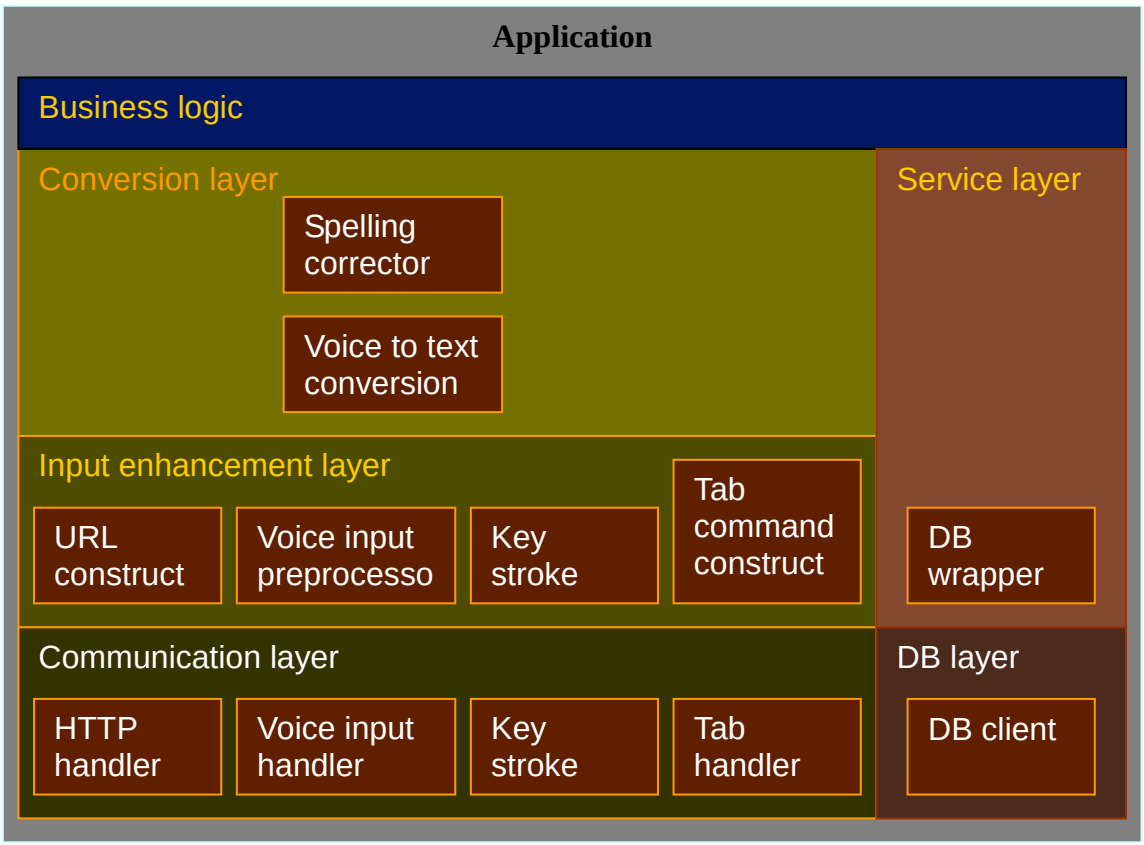

Figure 3.1 layers of the proposed solution

3.2 Communication layer

As shown in the figure 3.2, communication layer is mainly responsible for three types of communications such as reading sound signals from sound card via .NET libraries and NAudio library, invoke IE web browser via system libraries and HTTP communication with the web browser. C++ native methods in the **oleacc.dll** library and **user32.dll** library are used to invoke IE web browser. Implementations of the above functions are described in details in the implementation chapter. This layer is responsible for detecting registered key strokes and passes that to the above layers for execution of the programed logics.

In this project, we have used a content management system (CMS) to host information which will be searched via the proposed standalone application by using HTTP communications. These information will be displayed on a new IE tab. Implementation of the information searching via HTTP communication will be described in details in the implementation chapter.

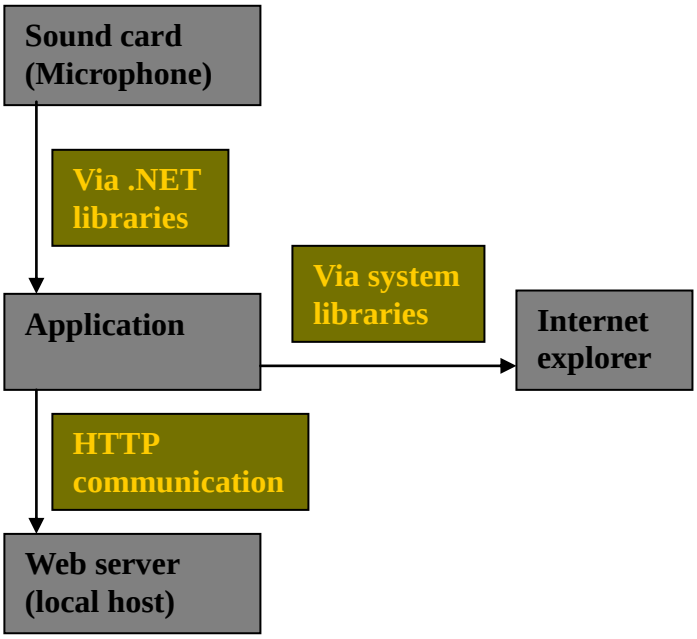

Figure 3.2 communication mechanism of the proposed solution

As previously explained, we are reading PCM 16 bit samples via .NET libraries and NAudio library. The sampled signal Xs can be represented as the product of the time signal and an impulse train, the sampling function (also called a comb function). Bellow equation will explain the time domain sampling mathematically.

$$x_s(t) = x(t) \cdot comb(t) = x(t) \cdot \sum_{m=-\infty}^{\infty} \delta(t - mt_0) = \sum_{m=-\infty}^{\infty} x[m]\delta(t - mt_0)$$

Its frequency domain representation is as follows.

$$X_s(j\omega) = \int_{-\infty}^{\infty} x_s(t)e^{-j\omega t}dt = \int_{-\infty}^{\infty} [\sum_{m=-\infty}^{\infty} x[m]\delta(t - mt_0)]e^{-j\omega t}dt$$

$$\sum_{m=-\infty}^{\infty} x[m] \int_{-\infty}^{\infty} \delta(t - mt_0)e^{-j\omega t}dt = \sum_{m=-\infty}^{\infty} x[m]e^{-j\omega mt_0} = \sum_{m=-\infty}^{\infty} x[m]e^{-j2\pi fmt_0}$$

3.3 Input enhancement layer

This layer is mainly responsible for filtering the sound input, recording filtered input and streaming sound signals. In this project, we have used a law pass time domain filter to remove background noise of the sound input. Bellow equation is describing the FIR filtering mathematically.

$$y(n)=h(n)\otimes x(n)=\sum_{k=0}^{N-1} h(k)x(n-k) = \sum_{k=0}^{N-1} a_k x(n-k)$$

Where **N** is the size of the filter and x(n−k)=0 if (n−k)<0

Bellow example will explain the calculation which will be executed by the code. In here a0, a1, a2, a3, a4 are constants during the filtering process, y is the output and x is the input signal.

Convolve **x** with **h** to get **y**.

$\mathbf{y}(n) = \mathbf{h}(n) \otimes \mathbf{x}(n)$

$h = [a_0, a_1, a_2, a_3, a_4]$

$\mathbf{y}(5) = a_4x_0 + a_3x_1 + a_2x_2 + a_1x_3 + a_0x_4$

filter coefficients

a_4	a_3	a_2	a_1	a_0
x_0	x_1	x_2	x_3	x_4

audio samples

x_5 x_6 x_7 x_8 ...

To compute **y**(6), slide **x** to the left.

While FIR filter is using forward filter only, IIR filter will use both forward filter and feedback filter parts. In this project, we are using an IIR filter for filtering the sound input by using NAudio library. Bellow equation is describing the IIR filtering mathematically.

$$y(n)=h(n)\otimes x(n)=\sum_{k=0}^{N-1} a_k x(n-k) - \sum_{k=1}^{M} b_k y(n-k)$$

Where **N** is the size of the forward filter, **M** is the size of the feedback filter, and x(n−k)=0 if n−k<0

Filtered sounds were recoded as a wav file in this project. This filtered sound will be the input for the next layer. Implementation of filtering and recording will be described in the implementation chapter in detail.

3.4 Conversion layer

This layer is mainly responsible for converting the filtered sound into texts, reject less

accurate recognitions and correct falsely identified words by predicting upon the previous recognitions. Bellow example will explain a two-state Markov chain.

$$P = \begin{pmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{pmatrix}$$

(α)is the probability of changing from state 1 to 2. (1-α) is the probability of staing at the same state. (β) is the probability of changing from state 2 to 1. Finally, (1-β) is the probability of staing at the same state.

As above example explained, in our project we have calculated the probabilities of occurring next word after a given word by providing training set initially. We have compared the recognized word from the recognition engine with the highest probable words and select the most matching word from the highest probable word list. To compare two words (strings), we have used the Levenshtein distance between two words.

Mathematically, the Levenshtein distance between two strings a, b (of length | a | and | b | respectively) is given by lev a, b (| a |, | b |) where,

$$\operatorname{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} \operatorname{lev}_{a,b}(i-1,j) + 1 \\ \operatorname{lev}_{a,b}(i,j-1) + 1 \\ \operatorname{lev}_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases}$$

where 1 (ai ≠ bj) is the indicator function equal to 0 when ai = bj and equal to 1 otherwise, and lev a , b (i , j) is the distance between the first i characters of a and the first j characters of b.

As above function explained, we have calculated the distance between the recognized word from the recognition engine with the highest probable words and select the lowest distance word from the probable words list.

3.5 Business logic layer

This layer is mainly responsible for clustering converted words and constructs the search query to retrieve results via HTTP communication. For clustering, we have used naïve bayes theory. Equation of the naïve bayes theory is explained bellow.

Bayes Theorem to predict the probability.

$P(h/D) = [P(D/h)\ P(h)]/P(D)$

Where,

P(h) : Prior probability of hypothesis h (predicted value)

P(D) : Prior probability of training data D (value given)

P(h/D) : Probability of h given D (expected o/p)

Above basic equation is able to cluster word list into two categories only. We have enhanced the naïve Bayes theory to cluster a word list into several categories. Implementation chapter will describe this algorithm in details.

3.6 Summery

In this chapter, we have presented the layers of the application and theories which are used in each layer in details. We have explained mainly sampling theory, FIR filtering theory, IIR filtering theory, Markov chain theory, Levenshtein distance calculation and Bayes Theorem in this chapter. Implementation of the above mentioned theories will be explained in the implementation chapter in details. In here, we have explained the programming languages used, developing environment, libraries used and required environment for the standalone application also.

Chapter4

Using voice to text conversion for efficient information searching

4.1 Introduction

Here we describe voice to text conversion for efficient information searching. Having define the problem in chapter2, we present the technology required for the propose solution in chapter3. This chapter presents our approach to use voice to text technology to address our problem. This approach is described under the headings of hypothesis, input to the system, output of the system, process to convert input to output, overall feature of the system and users.

4.2 Hypotheses

We can improve efficiency of information searching at call centers by introducing voice to text technology for fast inputs to the systems.

4.3 Input to the system

Customers will choose voice calls as a medium to contact the organization. Call center agents use headsets as the input device and a computer connected to intranet to answer customer queries. This voice input via sound card will be taken as the input to the software. Voice input will be preprocessed by using an audio library before it sends to the voice to text converting algorithm which is coded inside windows library. Converted texts are presented to user in the interface of the software to manipulate manually by using key inputs. Search command will be received by the operator of the software and out put will be displayed in a web browser.

4.4 Output of the system

Though the main output of the system is searched information with respect to a customer query, clustering of the input and predicting the level of customer query are also outputs of the system. Searched information will be presented on a web

browser. Searching criteria is displayed on the software as an intermediate output to the user.

4.5 Process

Figure 4.1 Overall processes of the system

As above diagram presents software contains three major processes namely convert speech to text, mining text and launch web browser with results. First process contains reading voice signal from the input, filtering the samples, enhancing the samples, converting enhanced voice signal to text and improving the outcome texts sub processes. Second process contains categorization of text which was the outcome of the first process and saving output for reporting purposes sub processes. Third process contains displaying results on web browser and focusing the results sub processes.

First process will receive the users input as a voice signal via microphone or headset. NAudio library which was written in C++ and capable of running on top of the .NET framework is used to read the input voice signal and it is in the format PCM16. Then it will be sent through a low pass filter to suppress the unwanted signals such as noise. This filter is a software filter which was written in C# on .NET framework. Then the filtered samples will be enhanced to send through the conversion process.

4.6 Overall features

In connection with the input, output, users and process, the overall features of the system include the following characteristics.

- Multiple inputs (voice input and agents input)
- User-friendly
- Research platform for call centers (analysis voice signals)

4.7 Users

There are number of users who can be benefited by the research such as call center agents who are accessing the system for information searching and managers who are accessing the system for reporting purposes. Those who are interested in studding voice to text algorithms can also use this system for learning purposes.

4.8 Summery

In this chapter, we have explained hypothesis, input, output, process, features and the users of the system. Process comprised with three sub processes namely speech conversion, categorization and show searched results in the web browser. This application was programed with C# and C++ programming languages and will provide a user friendly research platform for call centers.

Chapter5

Design

5.1 Introduction

Previous chapter gave full picture of the entire solution. This chapter describes the design of the solution presented in the approach. We have design the solution as a standalone system with HTTP client embedded to that for the communication between application and the backend. Here we describe the top-level architecture of the design by elaborating on the each component of the architecture.

5.2 Top-level architecture

The top-level architecture comprises of 3 main components namely reading and enhancing the voice signal and converting enhanced signal to text, categorizing texts and displaying searched results on web browser. Figure 5.1 illustrates the top-level architecture of the software implemented under this project. Upcoming sections will explain above mentioned main components one by one separately.

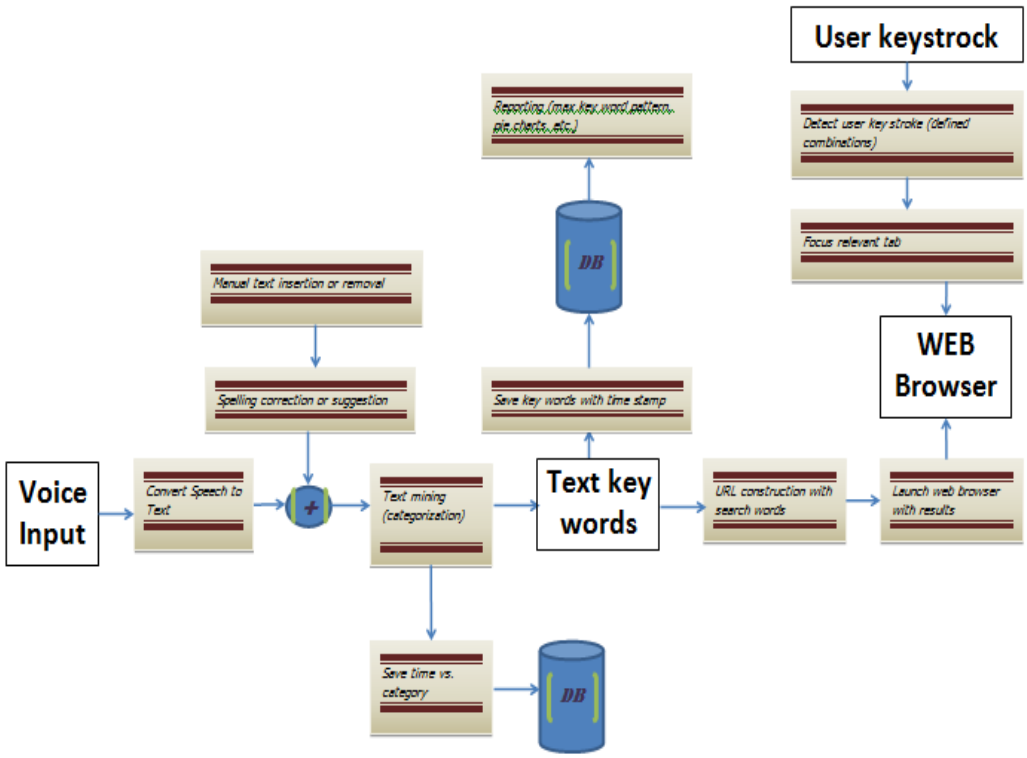

Figure 5.1 Top-level architecture of the system

5.3 Enhancing the voice signal and converting enhanced signal to text

This is the first step of the system and under this module we are discussing the reading inputs to the system, enhancing input, filtering input signal and converting enhanced voice signal to text. This module will read the voice signal of the agent as PCM16 signal from the sound card and filter it for high frequencies such as noise. Module uses **NAudio** open source library for reading and filtering the input. Filter is time domain low pass filter to remove background white noise from the input signal. Next chapter will explain the implementation of the filter and signal processing. At this stage it will create an intermediate wav file with the recoded sound. Then this wav file will be the input to the *voice to text* engine. The engine will convert the voice file into text by using loaded grammar to the engine. User of the system can manually edit the converted output or add extra details to the converted output. Resultant text of this module will be the input to the next module.

In addition to above paragraph, this component will reject less accurate recognitions and correct falsely identified words by predicting upon the previous recognitions. For

the predictions, we are using Markov chains and for the selection process we are using Levenshtein distance between predicted words and converted words.

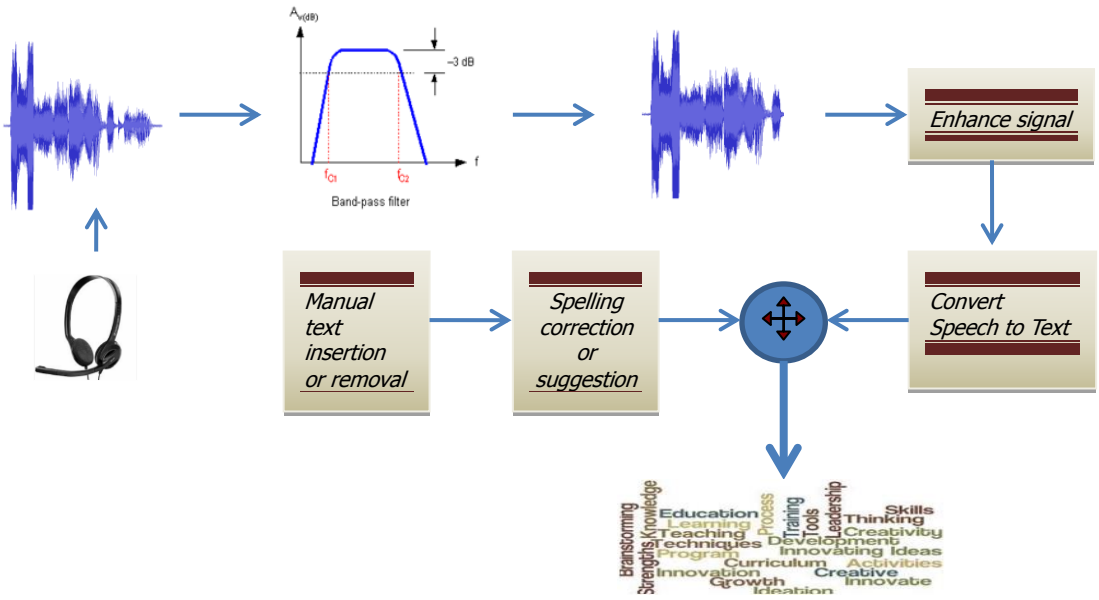

Figure 5.2 Architecture of the module reading and enhancing the voice signal

5.4 Categorizing texts

As described by section 5.3, first modules output will be the input to this module. This module will categorize the input text according to predefined metadata. For categorizing text into groups it will use naïve Bayes theory.

Naïve Bayes theory

P(h/D) = [P(D/h) P(h)]/P(D)

Where,

P(h) : Prior probability of hypothesis h (predicted value)

P(D) : Prior probability of training data D (value given)

P(h/D) : Probability of h given D (expected o/p)

Above Naïve Bayes theory was enhanced to categorize texts into several categories as explained bellow by using an example.

Example

ADSL: router, adsl, internet, web

PEO: router, peo, tv

BB: router, internet

Router key word belongs to three categories namely ADSL, PEO and BB. If the key word router appears as a result of voice to text conversion, it will show all three categories because of the same probability calculated. If the key word router and web appears as a result of voice to text conversion, it will show the category ADSL as a result. Next chapter will explain this further by using the code lines of the software and the library integration for this.

Then these calculated data can be saved for future reporting purposes such as investigating number of call per technical area against time. Output of this module is category of converted text and the keywords which are selected by the agent and the output of this module will be the input for next module which will be described under next section.

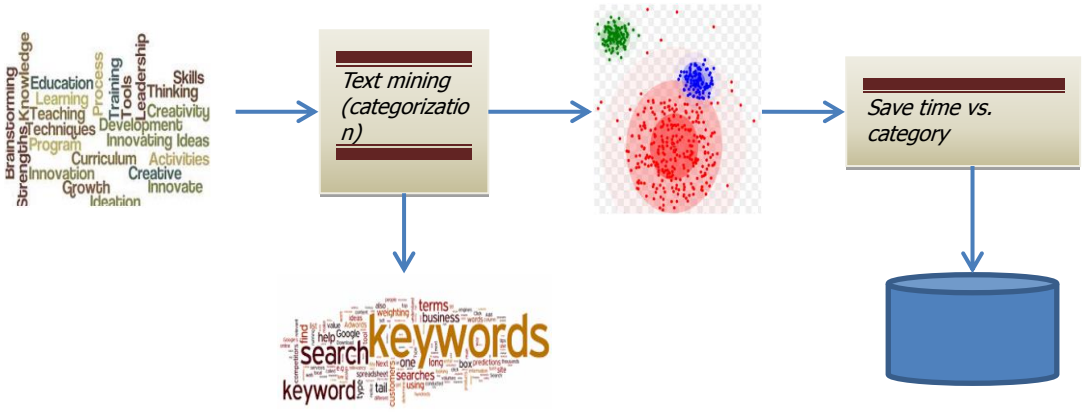

Figure 5.3 Architecture of the module categorizing texts

5.5 Displaying searched results on web browser

As described by section 5.4, second modules output will be the input to this module. This module is responsible for launching web browser with the results which are searched by the agents. These steps are presented as a flow diagram in figure 5.4. This module is capable of lounching default web browser with an URL. In this particular

scenario, we used wordpress content management system to host information which will be requested by the system via a HTTP call.

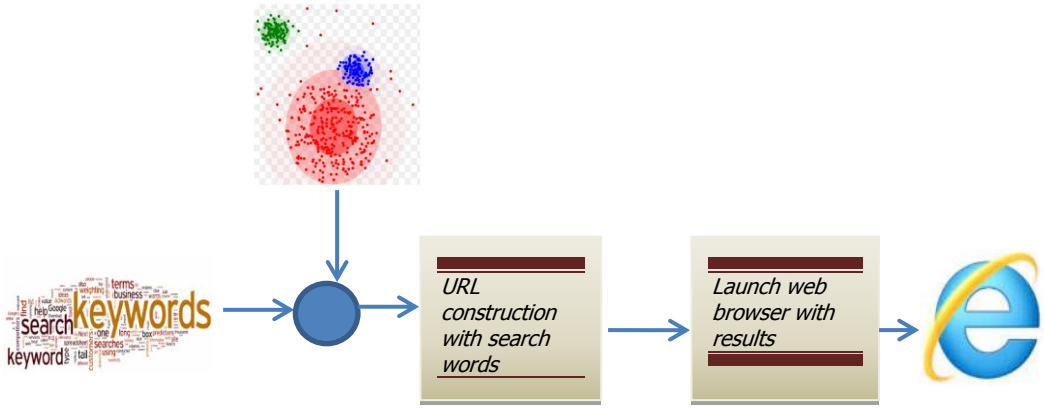

Figure 5.4 Architecture of the module displaying searched results on web browser

This module has the capability to focus IE tabs as a response to a key stroke. For this we need to access C++ native methods in the **oleacc.dll** library and **user32.dll** library. For an example, system can detect the key stroke SHIFT PLUS ONE and focus GMAIL opened window or tab in the IE web browser by enabling fast access to frequently used web sites. Implementation of the access of the above native method of **oleacc.dll** library and **user32.dll** library will be described in the next chapter more descriptively.

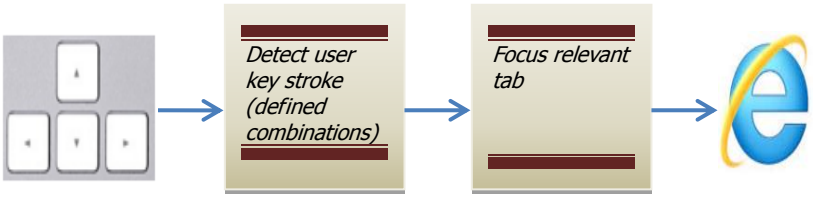

Figure 5.5 Focusing searched results

5.6 Summery

In this chapter, we have described the architecture of the application with the details of the main components namely reading and enhancing the voice signal and converting enhanced signal to text, categorizing texts and displaying searched results on web browser. Each component was described in details by using diagrams. Next chapter will explain the implementation of each component with details by using code segments.

Chapter6

Implementation

6.1 Introduction

Chapter 5 presented the design of the software and this chapter describes implementation of the software. Application was built in .NET4 framework and application has three modules namely enhancing the voice signal and converting enhanced signal to text module, categorizing texts and displaying searched results on web browser module which will be described separately in this chapter.

6.2 Overall solution

Overall solution has been implemented on .NET4 framework that can run on windows operating system. This is primarily standalone application with HTTP communication capability. The algorithms, hardware, software and relevant code segments of the implementation are presented in this chapter. Figure 6.1 shows the main window of the proposed solution.

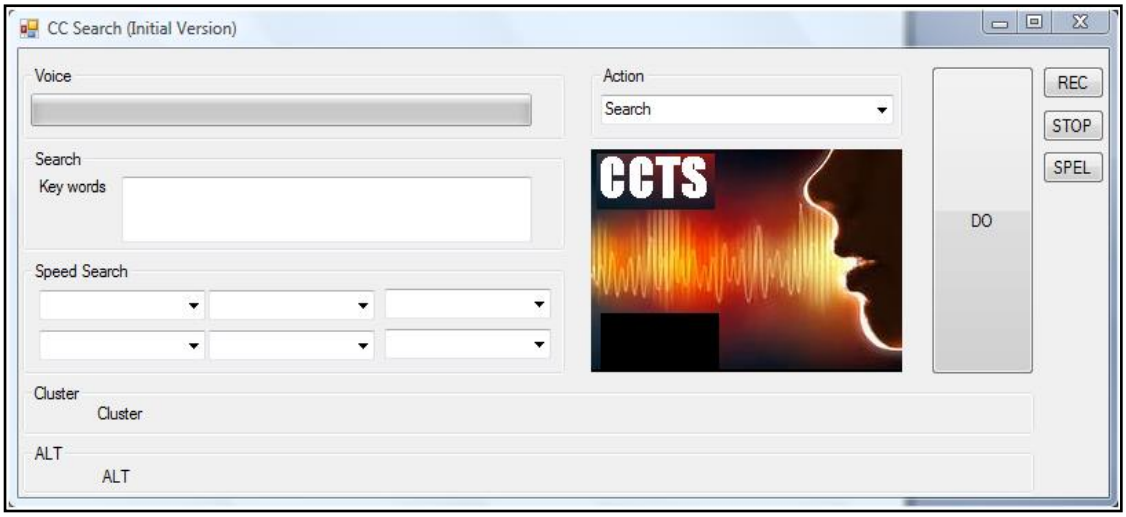

Figure 6.1 Main window of the software

6.3 Implementation of the module enhancing the voice signal and converting enhanced signal to text

As explained in the previous chapter, this module will read inputs of the system, enhance input, filter input signal and convert enhanced voice signal to text. In this chapter we are explaining the implementation of the module step by step. First step is reading the voice signal from the sound card of the computer. We are using a headset with an audio jack to implement and test the new solution. As described in the previous chapter this signal will be read as a PCM16 signal which means it has 16 bits to represent a signal sample. We are using **NAudio** library to read inputs from the sound card as explained. This library supports both native C++ and C# .NET technologies and we are using the C# .NET support functions of the library to implement our solution.

```
        waveSource = new WaveIn();
        waveSource.DataAvailable += new
EventHandler<WaveInEventArgs>(waveIn_DataAvailable);
        waveSource.RecordingStopped += new
EventHandler(waveSource_RecordingStopped);
        filter = BiQuadFilter.LowPassFilter(waveSource.WaveFormat.SampleRate, 10000, 1);
```

We need to initialize the library functions as shown in the above code segment. **WaveIn ()** function will selects the microphone as the input source for the program. Otherwise we have to specify the input source by using parameters of the **WaveIn** () function such as wav file input. In our solution, most important call back functions of the NAudio library are **waveIn_DataAvailable** and **waveSource_ RecordingStopped**. Implementation of the above two functions will present bellow. Fourth line is the low pass filter initialization. This filter is included in NAudio library and we need to set the cutoff values experimentally, because filtering important high frequencies may lead to unrecognizing of the voice spelled.

```
void waveIn_DataAvailable(object sender, WaveInEventArgs e)
{
        short g = 0;
        var floatBuffer = new List<float>();
        short sample = BitConverter.ToInt16(e.Buffer, index);
        for (int index = 0; index < e.BytesRecorded; index += 2)

                /**
```

```
                * Gain
                */
                float sample32 = (float)sample + g;
                sample32 /= (float)Int16.MaxValue;
                floatBuffer.Add(sample32);
            }

            /**
            * low pass filtering
            */
            for (int i = 0; i < floatBuffer.Count; i++)
            {
                waveFile.WriteSample(filter.Transform(floatBuffer[i]));
                waveFile.Flush();
            }
        }
```

Above code shows the override of the previously mentioned waveIn_DataAvailable NAudio function. When there is any input signal at the sound card, it will trigger this function. Because one sample is represented by 2bytes, we need to read the input to a short variable. But the filter takes sample arguments as float values. So we have to convert them back to IEEE float format for time domain filtering. This BiQuadFilter uses time domain filtering and it is faster than frequency domain filtering.

```
void waveSource_RecordingStopped(object sender, EventArgs e)
{
    if (waveSource != null)
    {
        waveSource.Dispose();
        waveSource = null;
    }
}
```

As explained in the previous chapter, we are creating an intermediate wav file by recoding the filtered sound signals. Above method will define the end of the wav file and trigger the reorganization engine to convert recoded sounds into text. So this is the end of enhancement and next we are explaining the conversion of the recoded signal.

```
// create the engine
speechRecognitionEngine = createSpeechEngine("en-US");

// hook to events
speechRecognitionEngine.SpeechRecognized += new
EventHandler<SpeechRecognizedEventArgs>(engine_SpeechRecognized);
speechRecognitionEngine.SpeechRecognitionRejected += new
EventHandler<SpeechRecognitionRejectedEventArgs>(recognizer_SpeechRecognitionReject
ed);
```

For recognizing recoded sounds, first we need to initialize the recognizing engine which is implemented with .NET library. We are using English language for this experiment. So we need to select language parameter as en-US. As above code segment, we need to implement two important methods of the recognizing engine as engine_SpeechRecognized and recognizer_SpeechRecognitionRejected. engine_SpeechRecognized method will be triggered at a successful recognition of a sound signal. We are checking the confidence of the recognized word prior to accept it. This will improve the quality of the resultant texts. recognizer_SpeechRecognitionRejected method is implemented for experimental purposes of this project.

This project uses two types of grammar for voice conversion namely default dictation and custom dictation.

```
// Create a default dictation grammar.
DictationGrammar defaultDictationGrammar = new DictationGrammar();
defaultDictationGrammar.Name = "default dictation";
defaultDictationGrammar.Enabled = true;
```

Above code segment shows the implementation of default dictation which has the capability of converting any word spelled by the user. In this case it will match the words against the default dictionary installed in the computer.

```
Choices texts = new Choices();
string[] lines = File.ReadAllLines(Environment.CurrentDirectory + "\\GRAM.txt");
foreach (string line in lines)
{
   // skip commentblocks and empty lines..
   if (line.StartsWith("--") || line == String.Empty) continue;
```

```
    // split the line
    var parts = line.Split(new char[] { '|' });

    // add the text to the known choices of speechengine
    texts.Add(parts[0]);
}

// Grammar
Grammar wordsList = new Grammar(new GrammarBuilder(texts));
speechRecognitionEngine.LoadGrammar(wordsList);
```

Unlike default dictation, above code will match spelled word against the configured wordlist. This word list will be provided as a text file argument. In addition to that there is word prediction logic to correct falsely identified words. Bellow code example will explain the initialization of that logic by using the initial training set.

```
string seed = "";
string[] lines = File.ReadAllLines(Environment.CurrentDirectory + "\\Markov.txt");
foreach (string line in lines)
{
    seed = line;
    break;
}

seed = Tidy(seed);
// tokenise the input string
var seedList = new List<string>(Split(seed.ToLower()));
// create a chain with a window size of 4
chain = new Chain<string>(seedList, 4);
```

The input for the above logic is a text file which contains a text paragraph. This logic will generate the Markov chain for the prediction logic. Each time a word was recognized, it will be compared with the Markov chain predicted words and match it to the closet word by using Levenshtein distance algorithm. Bellow code segment will explain the Levenshtein distance algorithm which was used in this project.

```
int sourceLength = source.Length;
int targetLength = target.Length;
int[,] distance = new int[sourceLength + 1, targetLength + 1];
```

```
// Step 1
for (int i = 0; i <= sourceLength; distance[i, 0] = i++) ;
for (int j = 0; j <= targetLength; distance[0, j] = j++) ;
for (int i = 1; i <= sourceLength; i++)
{
   for (int j = 1; j <= targetLength; j++)
   {
      // Step 2
      int cost = (target[j - 1] == source[i - 1]) ? 0 : 1;
      // Step 3
      distance[i, j] = Math.Min(Math.Min(distance[i - 1, j] + 1, distance[i, j - 1] + 1), distance[i -
1, j - 1] + cost);
   }
}
return distance[sourceLength, targetLength];
```

Next paragraph will explain the way software takes arguments from the user to manipulate the resultant texts prior to perform the search operation. This enhances the search query because of the multiple input sources.

```
using NetSpell.SpellChecker;
/// <summary>
/// SpellChecker
/// </summary>
Spelling spellChecker = new Spelling();
```

We are providing spelling correction and suggestions to user input to avoid unexpected results because of the wrong spellings. For that we are using NetSpell library as shown in the above code segment.

```
// add event handlers
        spellChecker.MisspelledWord += new
NetSpell.SpellChecker.Spelling.MisspelledWordEventHandler(SpellChecker_MisspelledWord
);
        spellChecker.EndOfText += new
NetSpell.SpellChecker.Spelling.EndOfTextEventHandler(SpellChecker_EndOfText);
        spellChecker.DoubledWord += new
NetSpell.SpellChecker.Spelling.DoubledWordEventHandler(SpellChecker_DoubledWord);
```

We need to implement 3 main functions namely SpellChecker_MisspelledWord, SpellChecker_EndOfText and SpellChecker_DoubledWord to provide the above

functionality. spellChecker.SpellCheck () method call will launch the screen shown in Figure 6.2 to the user. User can select the correct word from the provided word list or ignore the suggestion. On successful selection or ignore of the suggestion, it will trigger the method SpellChecker_EndOfText which is displayed bellow. We can implement our custom logic inside that method.

```
private void SpellChecker_EndOfText(object sender, System.EventArgs args)
{
        // update text
        string s = spellChecker.Text;
        textBox1.Text = textBox1.Text + "," + s;
}
```

Full code implementation of this module is attached in the appendix A and the low pass filter implementation is attached in the appendix B.

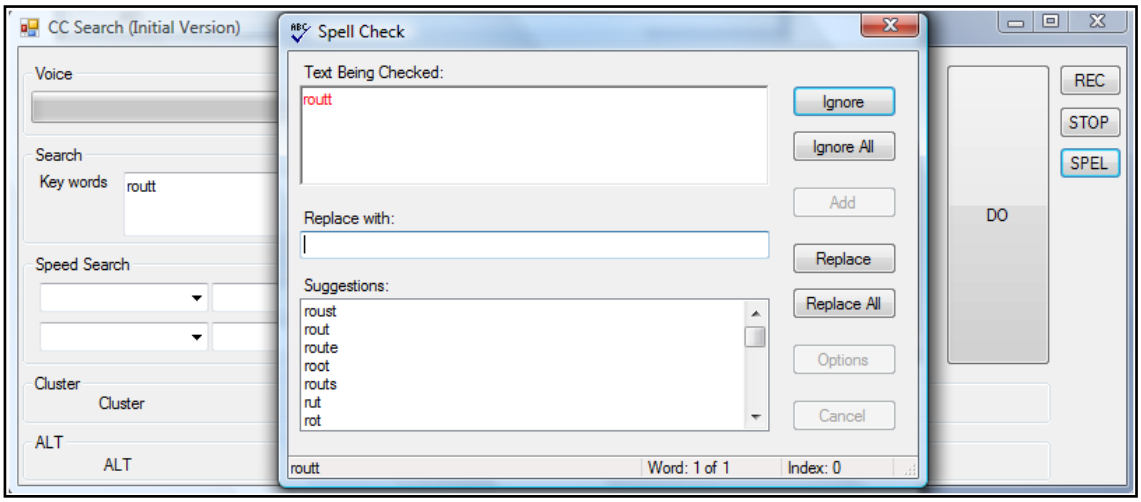

Figure 6.2 Spelling correction and suggestion window

6.4 Implementation of the module categorizing texts

As previously explained this module is responsible for categorizing the words which are converted from voice in the previous module. Theory behind this implementation is naïve bayes theory which was explained in previous chapter. Although naïve bayes theory categorize a word set into two groups namely positive and negative, enhanced implementation is capable of calculating the probability of more than two groups. Firstly we need to feed metadata into the calculation logic. This will be done via a text

file. For an example below text file content will create the data structure explained in Figure 6.3.

Example text file content

A:ADSL:router, adsl, internet, web

A:PEO:router, peo, tv

B:router:ZTE, DLINK

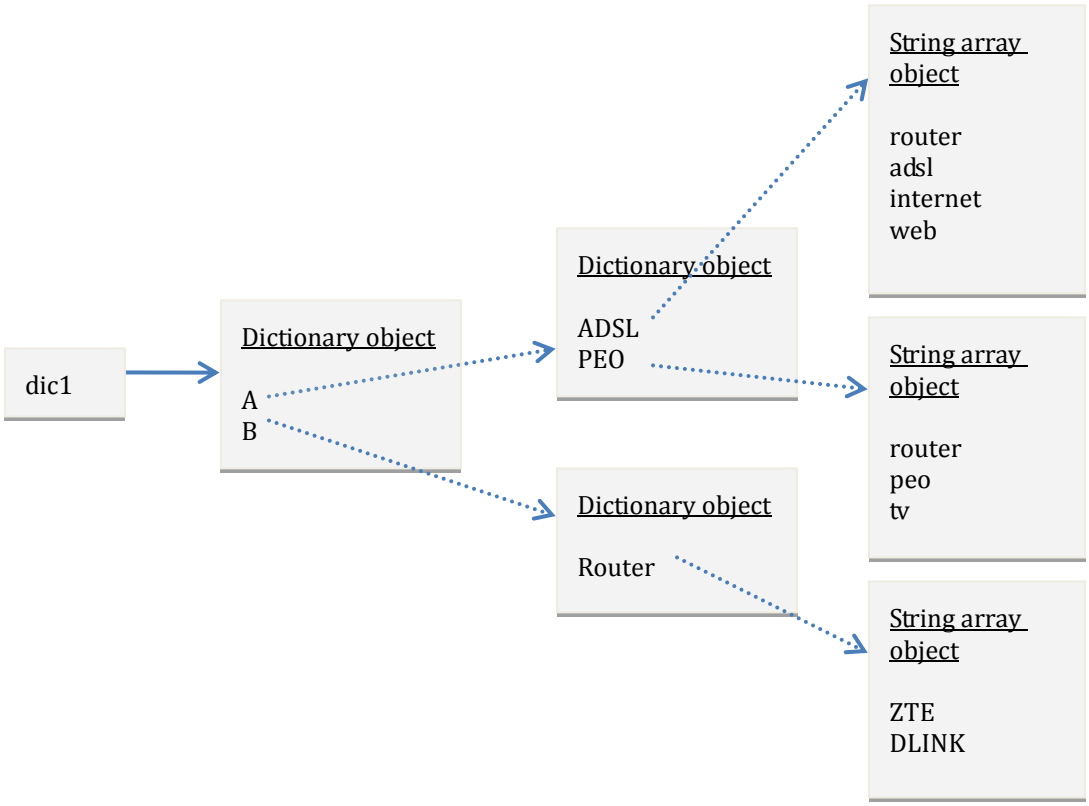

Figure 6.3 Sample data structure of cluster metadata

Below code segment will read the text file which was given as the input to the logic and create the above mentioned data structure. This data structure is the metadata to train the BayesClassifier which will explain later. After training the BayesClassifier, above data structure will be replaced with a collection of BayesClassifiers. This collection will be used to categorize the word list later.

```
string[] lines = File.ReadAllLines(Environment.CurrentDirectory + "\\Cluster.txt");
foreach (string line in lines)
{
        // split the line
        var parts = line.Split(new char[] { ':' });

        if (dic1.ContainsKey(parts[0]))
```

```
        {
            Dictionary<String, String[]> tdic = dic1[parts[0]];
            tdic[parts[1]] = parts[2].Split(new char[] { ',' });
        }
        else
        {
            Dictionary<String, String[]> tdic = new Dictionary<string, string[]>();
            tdic[parts[1]] = parts[2].Split(new char[] { ',' });
            dic1[parts[0]] = tdic;
        }
    }
```

BayesClassifier is the object which has the capability to categorize a given word into a category. In our scenario same word list will be categorized into several categories such as ADSL and router, PEO and router. We have implemented groups of category lists such as A, B, C, etc. because of the dynamic structure we can add any number of category list groups to the configuration file. Below code segment will create one BayesClassifier per category list and teach it. For an example, if we are using two category lists like A and B, it will create two BayesClassifiers.

```
/// teach BayesClassifier
foreach (KeyValuePair<String, Dictionary<String, String[]>> entry in dic1)
{
    BayesClassifier.Classifier m_Classifier1 = new BayesClassifier.Classifier();
    vm_Classifier.Add(m_Classifier1);
    Teach(entry.Value, m_Classifier1);
}

private void Teach(Dictionary<String, String[]> tdata, BayesClassifier.Classifier
m_Classifier1)
{
    foreach (KeyValuePair<String, String[]> entry in tdata)
    {
        m_Classifier1.TeachPhrases(entry.Key, entry.Value);
    }
}
```

Full code implementation of this module is attached in the appendix A and the BayesClassifier implementation is attached in the appendix C.

6.5 Implementation of the module Displaying searched results on web browser

As previously explained this module is responsible for launching web browser with the results which are searched by the agents. Below code segment will take the resultant text which is derived form voice to text conversion and the users input and construct a URL with a predefined prefix and launch the web browser with that URL.

```
string prefix = "http://172.25.37.187/wordpress/?s=";
string test = prefix + tabName.Replace(",", "+");
string test1 = test + "+" + comboBox2.Text + "+" + comboBox3.Text + "+" +
comboBox4.Text + "+" + comboBox5.Text + "+" + comboBox6.Text + "+" + comboBox7.Text;
Process.Start(test1);
```

Second important functionality of this module is handling predefined key strokes such as SHIFT PLUS ONE. For this purpose we have to use two system library methods namely RegisterHotKey and UnregisterHotKey. Below code segment shows the signature of those two system methods.

```
[DllImport("user32.dll")]
private static extern bool RegisterHotKey(IntPtr hWnd, int id, int fsModifiers, int vk);

[DllImport("user32.dll")]
private static extern bool UnregisterHotKey(IntPtr hWnd, int id);
```

Below code segment explains the way of registering a key combination such as SHIFT PLUS ONE. SHIFT is the constraint and the key is ONE. Both fields hold an integer value and uses Boolean operators to combine them. Then this combined value will register to receive triggers when user presses the above mention key combination.

```
public static class Constants
{
        //modifiers
        public const int NOMOD = 0x0000;
        public const int ALT = 0x0001;
        public const int CTRL = 0x0002;
        public const int SHIFT = 0x0004;
        public const int WIN = 0x0008;
}
ghk1 = new GlobalHotkey(Constants.SHIFT, Keys.D1, this);
```

When user presses the above registered key combination, following overridden method will be triggered with a parameter which contains the pressed key. We wrote our custom logic inside the method to perform key specific operations.

```
protected override void WndProc(ref Message m)
{
        if (m.Msg == Constants.WM_HOTKEY_MSG_ID)
        {
                switch (GetKey(m.LParam))
                {
                        case Keys.D1:
                        // the hotkey key is (Shift + 1)
                        HandleHotkey("1");
                        break;
                }
        }
}
```

Third important functionality of this module is the ability to focus a tab on IE web browser. For that we need access below two system methods.

```
[DllImport("oleacc.dll")]
private static extern int AccessibleObjectFromWindow(IntPtr hwnd, uint id, ref Guid iid, [In,
Out, MarshalAs(UnmanagedType.IUnknown)] ref object ppvObject);

[DllImport("oleacc.dll")]
private static extern int AccessibleChildren(IAccessible paccContainer, int iChildStart, int
cChildren, [In, Out, MarshalAs(UnmanagedType.LPArray, SizeParamIndex = 2)] object[]
rgvarChildren, out int pcObtained);
```

Below code segment explains the logic behind focusing a specific IE tab. Firstly we need to take a handler to currently running IE instance. Then it will retrieve a UI handler to that IE instance. Then we will iterate its child instances which mean tabs and retrieve its displaying name. That displaying name will match against the required name. If displaying name contains the required name, it will issue the activate command to focus that tab or child instance.

```
AccessibleObjectFromWindow(GetDirectUIHWND(ieHandle), OBJID.OBJID_WINDOW,
ref accessible);
if (accessible == null) throw new Exception();
IEAccessible ieDirectUIHWND = new IEAccessible(ieHandle);
```

```
foreach (IEAccessible accessor in ieDirectUIHWND.Children)
{
        foreach (IEAccessible child in accessor.Children)
        {
                foreach (IEAccessible tab in child.Children)
                {
                        if
                        (tab.Name.ToLower().Contains(tabCaptionToActivate.ToLower()))
                        {
                                tab.Activate();
                                return true;
                        }
                }
        }
}
```

Full code implementation of this module is attached in the appendix A and the IEAccessible implementation is attached in the appendix D.

6.6 Summery

In this chapter, implementations of the main three modules were explained with details namely enhancing the voice signal and converting enhanced signal to text module, categorizing texts and displaying searched results on web browser module. In addition to that, we have explained the implementations of the sampling theory, IIR filtering theory, Markov chain theory, Levenshtein distance calculation and Bayes Theorem which were used in above mentioned modules in this chapter. This chapter also explained about the external libraries and standard libraries which were used to implement the application such as .NET libraries, NAudio library and SpellChecker library.

Chapter7
Evaluation

7.1 Introduction

This chapter describe how we formally evaluated (tested) the proposed solution. In this sense, we explain test cases, participants used, testing setup, feedback received, data collection and analysis. Bellow test cases are the high level test cases of the proposed software and the results were attached as an appendix (appendix E) at the end of the thesis.

Major test cases
1. Enable recoding by clicking the recode button and spell a word like router. Then stop the recoding by clicking the stop button and verify whether the spelled word was converted to text correctly or not.

2. Enable recoding by clicking the recode button and spell a word like router. Then stop the recoding by clicking the stop button and verify whether the spelled word was recoded in the intermediate wav file without background noise or not.

3. Enable recoding by clicking the recode button and spell a word like router. Then stop the recoding by clicking the stop button. After that click do button by selecting search as the action. Verify whether it shows the correct cluster or not.

4. Enable recoding by clicking the recode button and spell a word like router. Then stop the recoding by clicking the stop button. After that click do button by selecting search as the action. Verify whether it shows searched results in the web browser or not.

When the application was launched main interface will be showed as figure 7.1. It shows voice signal fluctuation as a bar and the detected and converted texts. Those converted texts are editable to the user for the purpose of manual insertion of the extra text to enhance the search function. It will display the cluster of the converted words by using the naïve bays theory. It contains four buttons to start recoding the voice signal, stop recoding, and spell checking and proceeding with search functionality.

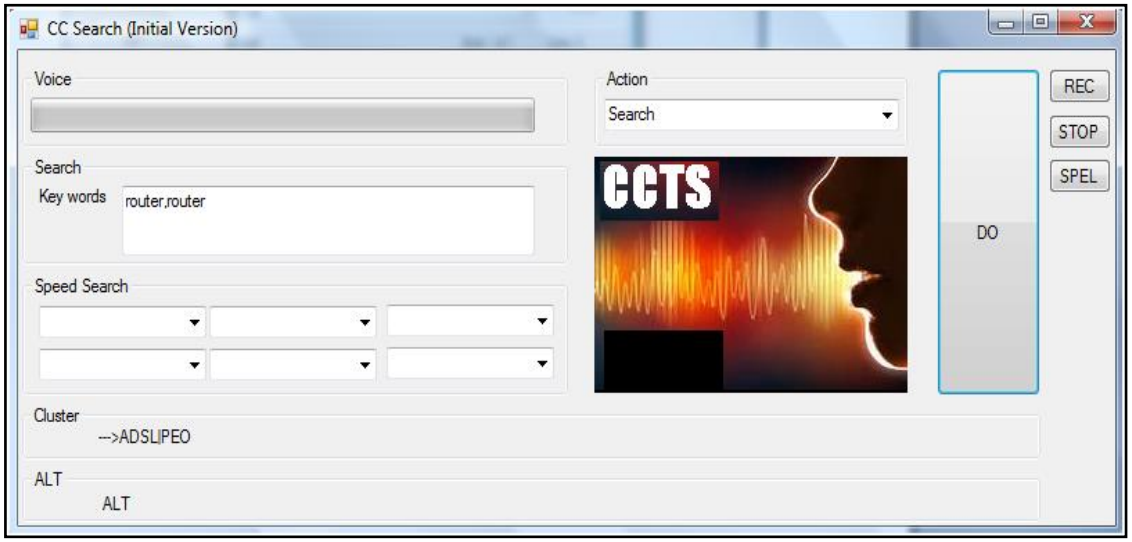

Figure 7.1 Main window with voice to text conversion results

7.2 Noise removal of the sound input

In this solution we used NAudio library for receiving sound input, noise removal (filtering input) and enhancing the voice signal. For the purpose of filtering noise which is included in the sound input, we used a low pass filter with the cutoff frequency of 10000 Hz. Bellow code segment shows the initial configurations of the filter which will be tested in this chapter.

filter = BiQuadFilter.LowPassFilter(waveSource.WaveFormat.SampleRate, 10000, 1);

For analyzing the recoded sound as wav file, we have used FFT analysis over the recorded sound. We have used math lab in built FFT analysis functions to get the FFT of the recorded wav file. Bellow code segment will present the math lab code which was used to analyze the file.

```
f='Test0001.wav';
[y,Fs,bits] = wavread(f);

Nsamps = length(y);
t = (1/Fs)*(1:Nsamps)          %Prepare time data for plot

%Do Fourier Transform
y_fft = abs(fft(y));           %Retain Magnitude
y_fft = y_fft(1:Nsamps/2);     %Discard Half of Points
f = Fs*(0:Nsamps/2-1)/Nsamps;  %Prepare freq data for plot
```

```
%Plot Sound File in Time Domain
figure
plot(t, y)
xlabel('Time (s)')
ylabel('Amplitude')
title('Time Domain')

%Plot Sound File in Frequency Domain
figure
plot(f, y_fft)
xlim([0 4000])
xlabel('Frequency (Hz)')
ylabel('Amplitude')
title('Frequency Response')
```

We have spelled the word ‘ONE’ and we have analyzed the recoded wav file by using the above mentioned math lab code segment. Figure 7.2 shows the UI of the application when we are executing this test case. It shows the recognized word as ‘ONE’.

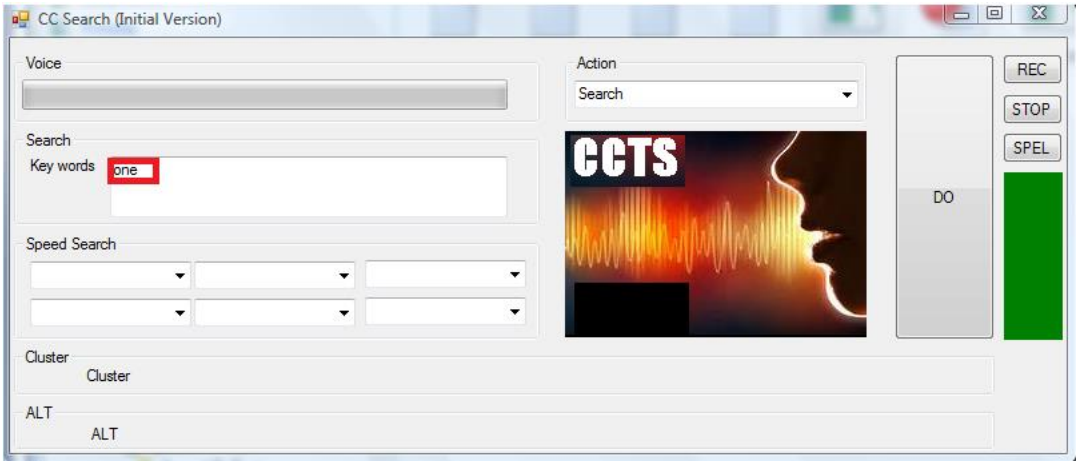

Figure 7.2 UI of the application in operation

As the output of the above code segment, bellow graphs can be observed for the recorded wav file. Figure 7.3 shows the time domain representation of the wav signal. Because we have recorded word ONE, it shows an amplitude fluctuation segment in the graph. Figure 7.4 show the frequency domain representation of the wav signal. Because of the law pass filter which was used with cutoff frequency 10000 Hz, it

shows no high frequencies in the graph. This verifies that the law pass filtering in the preprocessing module was given the expected results.

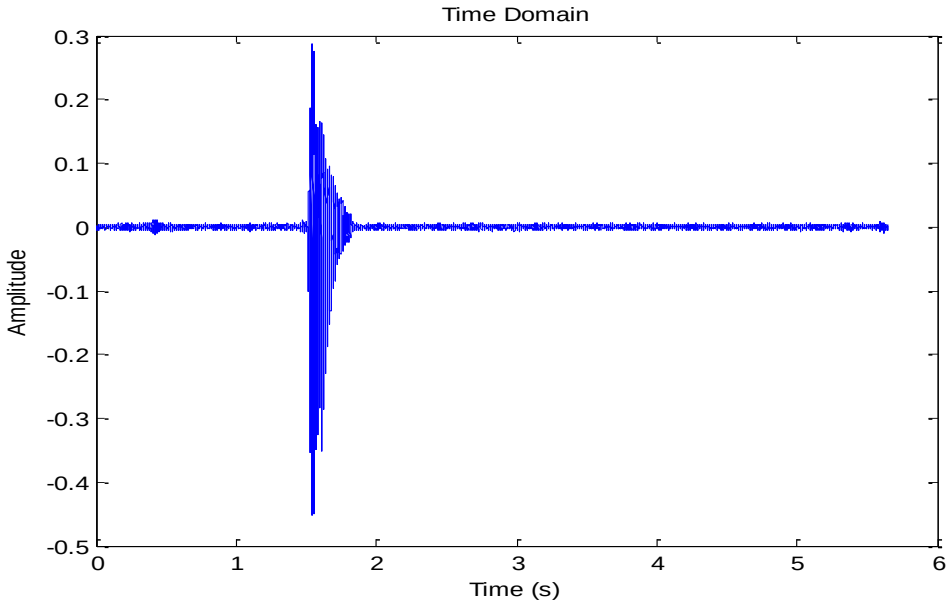

Figure 7.3 Time domain output of the wave file

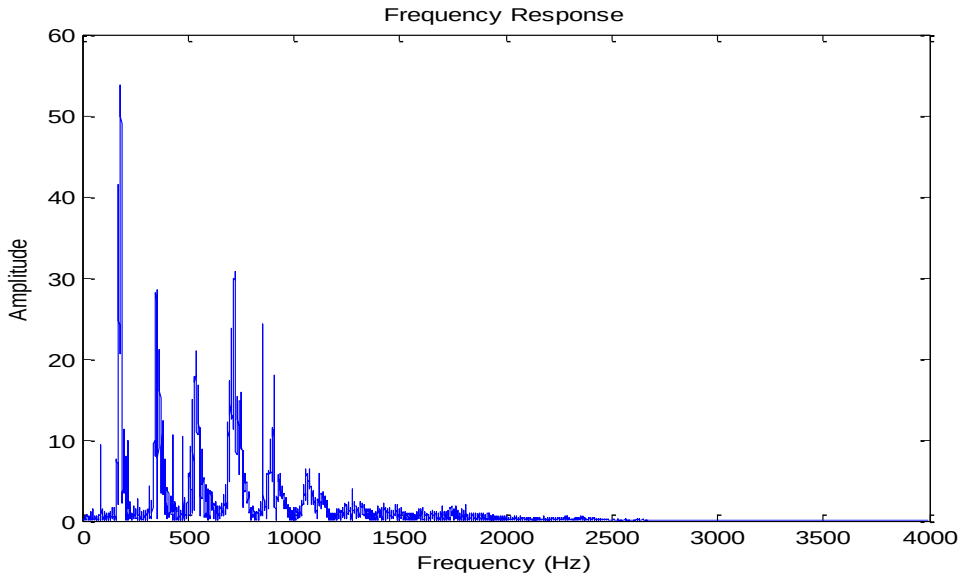

Figure 7.4 Frequency domain output of the wave file

7.3 Test cases

Test cases at the appendix E were tested by using multiple users and it gave successful results for the conducted test cases. These test cases measured the functionality of the developed application.

7.4 Performance test

These test cases tested the accuracy and the performance of the developed application by using multiple users. Bellow sentences were spelled by multiple users and the results were presented as a table bellow.

1. *internet,is,slow*
2. *internet,browsing,is,slow*
3. *new,connections,request*
4. *mail,box,is,full*
5. *order,a,television,channel*
6. *bill,is,not,receiving*

Bellow paragraph show the training data set for the Markov model.

internet is not working. internet browsing is slow. internet usage limit. need to reset password. need to create portal account. television program is not available. no television screen display. order a television channel. new connections request. bill is not receiving. mail box is full. mail is not receiving. unwanted mails.

Bellow paragraph shows the starting words for the training data set for the Markov model.

Internet, need, no, order, new, bill, mail, unwanted, television

Bellow paragraph shows the Cluster metadata for information clustering.

A:ADSL:internet, adsl, web, slow

A:PEO:internet, tv, television, channel

A:PSTN:new, connections, telephone

A:EMAIL:mail, box, receiving

A:BILL:bill, receiving

User	Spelled sentences	Comments (speech to text)	Cluster	Information searching (results are displayed correctly)
1	1. internet,is,slow 2. internet,browsing,is,slow 3. new,connections,request 4. mail,box,is,full 5. order,a,television,channel 6. bill,is,not,receiving	*Internet* word sometimes not recognized. *Bill* word did not identify.	1. ADSL 2. ADSL 3. PSTN 4. EMAIL 5. PEO 6. BILL	1. Yes 2. Yes 3. Yes 4. Yes 5. Yes 6. Yes
2	1. internet,is,slow 2. internet,browsing,is,slow 3. new,connections,request 4. mail,box,is,full 5. order,a,television,channel 6. bill,is,not,receiving	*New* word sometimes not recognized. *Bill* word did not identify.	1. ADSL 2. ADSL 3. PSTN 4. EMAIL 5. PEO 6. BILL	1. Yes 2. Yes 3. Yes 4. Yes 5. Yes 6. Yes
3	1. internet,is,slow 2. internet,browsing,is,slow 3. new,connections,request 4. mail,box,is,full 5. order,a,television,channel 6. bill,is,not,receiving	*Bill* word did not identify.	1. ADSL 2. ADSL 3. PSTN 4. EMAIL 5. PEO 6. BILL	1. Yes 2. Yes 3. Yes 4. Yes 5. Yes 6. Yes
4	1. internet,is,slow 2. internet,browsing,is,slow 3. new,connections,request 4. mail,box,is,full 5. order,a,television,channel 6. bill,is,not,receiving	*Internet* word and *New* word sometimes not recognized. *Bill* word did not identify.	1. ADSL 2. ADSL 3. PSTN 4. EMAIL 5. PEO 6. BILL	1. Yes 2. Yes 3. Yes 4. Yes 5. Yes 6. Yes

Table 7.1 Result of the performance test

Above table shows the results/comments of the performance test of the developed application. It shows that speech to text conversion performance is varying person to person and it depends on the words collection (Markov model metadata) which will be used to match with. It shows that information categorization and searching was functioned 100% accuracy.

7.5 Summery

This chapter presented the functionality of the UI of the application and the test cases which were used to test the application developed. According to the conducted test cases application meets the requirements of the proposed solution. And also we have presented the evaluation results of the preprocessing stage of the voice to text module. According to those results, filtering stage was success and delivered the expected results from the employed law pass filter.

Chapter8

Conclusion

8.1 Introduction

We have conduct the research to reduce the queues for services and increase the customer satisfaction by reducing the service delay by using voice to text technology. This research reduced the thinking time and key board entering time by using voice to text as the main technology. Other than that we have used several technologies namely capturing users' key strokes, information categorization and focusing web browser's information as a response to a keystroke and several theories mainly sampling theory, FIR filtering theory, IIR filtering theory, Markov chain theory, Levenshtein distance calculation and Bayes Theorem to enhance the out put of the research.

8.2 Conclusion

Our research was success with the accuracy of information categorization 90% and 100% of information querieng from content management system via HTTP communication and displays it on the web browser. Although the voice to text conversion accuracy is varying with the sound cards performance, our prediction algorithm and signal enhancing algorithm increased the accuracy of the voice to text conversion than the normal voice to text conversion by using a voice to text engine.

8.3 Further work

Our application was successful with the single user's profile on a single machine and it is needed to improve the algorithm for multiple users in a single machine and same user for multiple machines. It is suggested that to implement an algorithm to check the level of suitability of the Markov model metadata and suggest alternatives to the metadata collection.

8.4 Summery

This chapter provided the conclusion of the overall research as success and suggests the further work and explains barriers of the project.

Reference

Becheikh, N., Ziam, S., Idrissi, O., Castonguay, Y., Landry, R., 2010. How to improve knowledge transfer strategies and practices in education? Answers from a systematic literature review. Res. High. Educ. J. 7, 1.

Dabrowski, M., 2013. Business Intelligence In Call Centers. Int. J. Comput. Inf. Technol. 2.

DobriSek, S., Gros, J., Mihelic, F., Pavesic, N., 1999. HOMER: a voice-driven text-to-speech system for the blind, in: Industrial Electronics, 1999. ISIE'99. Proceedings of the IEEE International Symposium on. IEEE, pp. 205–208.

Evgeniou, T., Cartwright, P., 2005. Barriers to Information Management. Eur. Manag. J. 23, 293–299. doi:10.1016/j.emj.2005.04.007

Fitzsimmons, J., 2005. Service Management: Operations, Strategy, Information Technology with Student CD, 5 edition. ed. McGraw-Hill/Irwin, Boston.

Gans, N., Koole, G., Mandelbaum, A., 2003. Telephone call centers: Tutorial, review, and research prospects. Manuf. Serv. Oper. Manag. 5, 79–141.

Garcia, D., Archer, T., Moradi, S., Ghiabi, B., 2012. Waiting in Vain: Managing Time and Customer Satisfaction at Call Centers. Psychology 3, 213–216. doi:10.4236/psych.2012.32030

Khan, R.A., Quadri, S.M.K., 2012. Business intelligence: an integrated approach. Bus. Intell. J. 5, 64–70.

Kotsovos, A., Kriemadis, A., n.d. CUSTOMER RELATIONSHIP MANAGEMENT (CRM) ON THE INTERNET-EVIDENCE FROM THE FOOTBALL SECTOR.

Kumar, A., Telang, R., 2012. Does the Web Reduce Customer Service Cost? Empirical Evidence from a Call Center. Inf. Syst. Res. 23, 721–737. doi:10.1287/isre.1110.0390

L. Michelle Bobbitt, Pratibha A. Dabholkar, 2001. Integrating attitudinal theories to understand and predict use of technology-based self-service: The Internet as an illustration. Int. J. Serv. Ind. Manag. 12, 423–450. doi:10.1108/EUM0000000006092

Larsson, A.O., Hrastinski, S., 2011. Blogs and blogging: Current trends and future directions. First Monday 16.

Lee, C.-S., Lee, C.C., Kwon, H.-B., 2007. A Framework of Outsourcing Decision-Making for Human Resource Information Systems. MIS Res. Relev. IT Pract. Korea Soc. Manag. Inf. Syst. Seoul Korea 1–6.

Lee, L.–., Oun-Young, M., 1988. Voice and text messaging-a concept to integrate the services of telephone and data networks, in: Communications, 1988. ICC'88. Digital Technology-Spanning the Universe. Conference Record., IEEE International Conference on. IEEE, pp. 408–412.

McGuire, K.A., Kimes, S.E., Lynn, M., Pullman, M.E., Lloyd, R.C., 2010. A framework for evaluating the customer wait experience. J. Serv. Manag. 21, 269–290.

Medhat, W., Hassan, A., Korashy, H., 2014. Sentiment analysis algorithms and applications: A survey. Ain Shams Eng. J. 5, 1093–1113. doi:10.1016/j.asej.2014.04.011

Meuter, M.L., Bitner, M.J., Ostrom, A.L., Brown, S.W., 2005. Choosing among alternative service delivery modes: An investigation of customer trial of self-service technologies. J. Mark. 69, 61–83.

Mozaheb, A., Alamolhodaei, S.M.A., Ardakani, M.F., others, 2015. Effect of customer relationship management (CRM) on performance of small-medium sized enterprises (SMEs) using structural equations model (SEM). Int. J. Acad. Res. Account. Finance Manag. Sci. 5, 42–52.

Oodith, D., Parumasur, S.B., 2014. Technology in a call center: an asset to managing customers and their needs? Probl. Perspect. Manag. 12.

Rasooli, P., Albadvi, A., 2007. Knowledge Management in Call Centres. Electron. J. Knowl. Manag. 5, 323–332.

Sabherwal, R., Becerra-Fernandez, I., 2011. Business intelligence : practices, technologies and management. Hoboken, NJ : Wiley.

Tsoukas, H., Vladimirou, E., 2001. What is Organizational Knowledge? J. Manag. Stud. 38, 973–993. doi:10.1111/1467-6486.00268

Tuerk, C., Monaco, P., Robinson, T., 1991. The development of a connectionist multiple-voice text-to-speech system, in: Acoustics, Speech, and Signal Processing, 1991. ICASSP-91., 1991 International Conference on. IEEE, pp. 749–752.

Appendix A

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Speech.Recognition;
using SpeechToText;
using System.IO;
using System.Diagnostics;
using System.Runtime.InteropServices;
using ADDIProtocol;
using NAudio.Wave;
using System.Threading;
using System.Collections;
using NetSpell.SpellChecker;
using System.Text.RegularExpressions;
using Markov;

namespace MSC
{
   public partial class Form1 : Form
   {

      #region locals

      /// <summary>
      /// data base util
      /// </summary>
      DB d = new DB();

      /// <summary>
      /// Markov Chain
      /// </summary>
      Chain<string> chain = null;

      /// <summary>
      /// Markov Chain initial words
      /// </summary>
      ArrayList chainstart = new ArrayList();

      /// <summary>
      /// prev recognized word from Markov Chain
      /// </summary>
      string prevrecognized = "";

      /// <summary>
      /// debug flag
      /// </summary>
      bool debug = true;

      /// <summary>
      /// Sites list for hot keys
      /// </summary>
      ArrayList sites = new ArrayList();

      /// <summary>
      /// SpellChecker
```

```
/// </summary>
Spelling spellChecker = new Spelling();

/// <summary>
/// Clustering
/// </summary>
BayesClassifier.Classifier m_Classifier = new BayesClassifier.Classifier();
ArrayList vm_Classifier = new ArrayList();

/// <summary>
/// NAudio source
/// </summary>
private WaveIn waveSource = null;

/// <summary>
/// LowPass filter
/// </summary>
BiQuadFilter filter = null;

/// <summary>
/// WAV file writer
/// </summary>
public WaveFileWriter waveFile = null;

/// <summary>
/// the engine
/// </summary>
SpeechRecognitionEngine speechRecognitionEngine = null;

/// <summary>
/// CMDs
/// </summary>
Dictionary<string, string> dic = new Dictionary<string, string>();

/// <summary>
/// Key Registration
/// </summary>
private GlobalHotkey ghk1;
private GlobalHotkey ghk2;
private GlobalHotkey ghk3;
private GlobalHotkey ghk4;
private GlobalHotkey ghk5;
private GlobalHotkey ghk6;

#endregion

#region invoke methods
[DllImport("user32.dll")]
static extern bool SetForegroundWindow(IntPtr hWnd);

[DllImport("user32.dll")]
private static extern int ShowWindow(IntPtr hwnd, int nCmdShow);

[DllImport("user32.DLL", CharSet = CharSet.Unicode)]
public static extern IntPtr FindWindow(String lpClassName, String lpWindowName);

private const int SW_HIDE = 3;
#endregion

#region constructer
/// <summary>
/// Constructer
/// </summary>
```

```
        public Form1()
        {

            InitializeComponent();
            Populatecluster();
            loadSites();
            LoadMarkov();

            try
            {
                // create the engine
                speechRecognitionEngine = createSpeechEngine("en-US");

                // hook to events
                speechRecognitionEngine.AudioLevelUpdated += new
EventHandler<AudioLevelUpdatedEventArgs>(engine_AudioLevelUpdated);
                speechRecognitionEngine.SpeechRecognized += new
EventHandler<SpeechRecognizedEventArgs>(engine_SpeechRecognized);
                speechRecognitionEngine.SpeechRecognitionRejected += new
EventHandler<SpeechRecognitionRejectedEventArgs>(recognizer_SpeechRecognitionRejected);

                // load dictionary
                loadGrammarAndCommands();
                loadCommands();

                if (debug)
                {
                    // use the system's default microphone
                    speechRecognitionEngine.SetInputToDefaultAudioDevice();
                    // start listening
                    speechRecognitionEngine.RecognizeAsync(RecognizeMode.Multiple);
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, "Voice recognition failed");
            }

            ghk1 = new GlobalHotkey(Constants.SHIFT, Keys.D1, this);
            ghk2 = new GlobalHotkey(Constants.SHIFT, Keys.D2, this);
            ghk3 = new GlobalHotkey(Constants.SHIFT, Keys.A, this);
            ghk4 = new GlobalHotkey(Constants.SHIFT, Keys.S, this);
            ghk5 = new GlobalHotkey(Constants.SHIFT, Keys.D, this);
            ghk6 = new GlobalHotkey(Constants.SHIFT, Keys.D3, this);

            try
            {
                // add event handlers
                spellChecker.MisspelledWord += new
NetSpell.SpellChecker.Spelling.MisspelledWordEventHandler(SpellChecker_MisspelledWord);
                spellChecker.EndOfText += new
NetSpell.SpellChecker.Spelling.EndOfTextEventHandler(SpellChecker_EndOfText);
                spellChecker.DoubledWord += new
NetSpell.SpellChecker.Spelling.DoubledWordEventHandler(SpellChecker_DoubledWord);
            }
            catch (Exception e)
            {

            }
        }

        #endregion
```

```
        #region internal functions and methods for speech recognition

        /// <summary>
        /// Creates the speech engine.
        /// </summary>
        /// <param name="preferredCulture">The preferred culture.</param>
        /// <returns></returns>
        private SpeechRecognitionEngine createSpeechEngine(string preferredCulture)
        {
            foreach (RecognizerInfo config in SpeechRecognitionEngine.InstalledRecognizers())
            {
                if (config.Culture.ToString() == preferredCulture)
                {
                    speechRecognitionEngine = new SpeechRecognitionEngine(config);
                    break;
                }
            }

            // if the desired culture is not found, then load default
            if (speechRecognitionEngine == null)
            {
                MessageBox.Show("The desired culture is not installed on this machine, the speech-engine will
continue using "
                    + SpeechRecognitionEngine.InstalledRecognizers()[0].Culture.ToString() + " as the default
culture.",
                    "Culture " + preferredCulture + " not found!");
                speechRecognitionEngine = new
SpeechRecognitionEngine(SpeechRecognitionEngine.InstalledRecognizers()[0]);
            }

            return speechRecognitionEngine;
        }


        /// <summary>
        /// Loads the commands.
        /// </summary>
        private void loadCommands()
        {
            try
            {
                string[] lines = File.ReadAllLines(Environment.CurrentDirectory + "\\CMD.txt");
                foreach (string line in lines)
                {
                    // split the line
                    var parts = line.Split(new char[] { '|' });
                    dic[parts[0]] = parts[1];
                }
            }
            catch (Exception ex)
            {
                throw ex;
            }
        }

        /// <summary>
        /// Loads the grammar and commands.
        /// </summary>
        private void loadGrammarAndCommands()
        {
            try
            {
                Choices texts = new Choices();
```

```
            string[] lines = File.ReadAllLines(Environment.CurrentDirectory + "\\GRAM.txt");
            foreach (string line in lines)
            {
               // skip commentblocks and empty lines..
               if (line.StartsWith("--") || line == String.Empty) continue;

               // split the line
               var parts = line.Split(new char[] { '|' });

               // add the text to the known choices of speechengine
               texts.Add(parts[0]);
            }

            // Grammar
            Grammar wordsList = new Grammar(new GrammarBuilder(texts));
            speechRecognitionEngine.LoadGrammar(wordsList);

            // Create a default dictation grammar.
            DictationGrammar defaultDictationGrammar = new DictationGrammar();
            defaultDictationGrammar.Name = "default dictation";
            defaultDictationGrammar.Enabled = true;

            // Create the spelling dictation grammar.
            //DictationGrammar spellingDictationGrammar = new
DictationGrammar("grammar:dictation#spelling");
            //spellingDictationGrammar.Name = "spelling dictation";
            //spellingDictationGrammar.Enabled = true;

            //// Create the question dictation grammar.
            //DictationGrammar customDictationGrammar = new DictationGrammar("grammar:dictation");
            //customDictationGrammar.Name = "question dictation";
            //customDictationGrammar.Enabled = true;

            speechRecognitionEngine.LoadGrammar(defaultDictationGrammar);
            //speechRecognitionEngine.LoadGrammar(spellingDictationGrammar);
            //speechRecognitionEngine.LoadGrammar(customDictationGrammar);
         }
         catch (Exception ex)
         {
            throw ex;
         }
      }

      #endregion

      #region speechEngine events

      /// <summary>
      /// Handles the SpeechRecognized event of the engine control.
      /// </summary>
      /// <param name="sender">The source of the event.</param>
      /// <param name="e">The <see cref="System.Speech.Recognition.SpeechRecognizedEventArgs"/>
instance containing the event data.</param>
      void engine_SpeechRecognized(object sender, SpeechRecognizedEventArgs e)
      {
         // This is for removing background noises
         if (e.Result.Confidence >= 0.2)
         {
            // recognized
            string recognized = e.Result.Text;
            string recognized1 = recognized;
            label3.Text = recognized;
```

```
            if (!textBox1.Text.Trim().Equals(""))
            {
               // generate a new sequence using a starting word, and maximum return size
               List<string> generated = new List<string>(chain.Generate(prevrecognized, 5));
               IEnumerable<string> t = chain.Generate1(prevrecognized, 3);
               int max = -1;

               for (int i = 0; i < t.Count(); i++)
               {
                  string ww = t.ElementAt(i);
                  double distance = CalculateSimilarity(ww, recognized);
                  int dis = (int)(distance * 100);
                  if (max < dis)
                  {
                     recognized1 = ww;
                     max = dis;
                  }
               }
            }
            else
            {
               // initial words
               int max = -1;
               for (int i = 0; i < chainstart.Count; i++)
               {
                  string ww = (string)chainstart[i];
                  double distance = CalculateSimilarity(ww, recognized);
                  int dis = (int)(distance * 100);
                  if (max < dis)
                  {
                     recognized1 = ww;
                     max = dis;
                  }
               }
            }

            prevrecognized = recognized1;
            textBox1.Text = (textBox1.Text.Trim() == "") ? recognized1 : textBox1.Text + "," + recognized1;
         }
      }

      /// <summary>
      /// Handles the AudioLevelUpdated event of the engine control.
      /// </summary>
      /// <param name="sender">The source of the event.</param>
      /// <param name="e">The <see cref="System.Speech.Recognition.AudioLevelUpdatedEventArgs"/>
instance containing the event data.</param>
      void engine_AudioLevelUpdated(object sender, AudioLevelUpdatedEventArgs e)
      {
         progressBar1.Value = e.AudioLevel;
      }

      /// <summary>
      /// Handle rejected words
      /// </summary>
      /// <param name="sender"></param>
      /// <param name="e"></param>
      void recognizer_SpeechRecognitionRejected(object sender, SpeechRecognitionRejectedEventArgs e)
      {
         string temp = "";
         foreach (RecognizedPhrase phrase in e.Result.Alternates)
         {
            temp = temp + "," + phrase.Text;
```

```
    }
    label3.Text = temp;
}



#endregion

#region HandleHotkey
/// <summary>
/// Execution of the CMD
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void button1_Click(object sender, EventArgs e)
{

    // Classify data ------------------
    String tabName2 = textBox1.Text;
    String data = tabName2.Replace(",", " ");
    String temp = "";
    foreach (BayesClassifier.Classifier value in vm_Classifier)
    {
        temp = temp + "--->" + Classify(data, value);
    }
    label2.Text = temp;

    // Add to SQL DB for future reporting purposes
    d.Add(temp + "###" + DateTime.Now);
    // --------------------------------

    string cmd = comboBox1.Text;
    String tabName1 = textBox1.Text;
    String tabName = tabName1;
    if (dic.ContainsKey(tabName1))
    {
        tabName = dic[tabName1];
    }

    if (cmd.Equals("Execute"))
    {
        // Action
        IEAccessible ie = new IEAccessible();

        Process[] processes = Process.GetProcessesByName("iexplore");
        foreach (var item in processes)
        {
            ShowWindow(item.MainWindowHandle, SW_HIDE);
            SetForegroundWindow(item.MainWindowHandle);
        }

        // --
        SHDocVw.ShellWindows shellWindows = new SHDocVw.ShellWindows();
        foreach (SHDocVw.InternetExplorer window in shellWindows)
        {
            ie.Focus((IntPtr)window.HWND, tabName);
        }
        // --
    }
    if (cmd.Equals("Search"))
    {
        string prefix = "http://172.25.37.187/wordpress/?s=";
        string test = prefix + tabName.Replace(",", "+");
```

```
            string test1 = test + "+" + comboBox2.Text + "+" + comboBox3.Text + "+" + comboBox4.Text +
"+" + comboBox5.Text + "+" + comboBox6.Text + "+" + comboBox7.Text;
            Process.Start(test1);
        }
    }

    /// <summary>
    /// Handle Hot key events
    /// </summary>
    /// <param name="key"></param>
    private void HandleHotkey(String key)
    {
        switch (key)
        {
            case "1":
                if (this.WindowState == FormWindowState.Minimized)
                {
                    this.WindowState = FormWindowState.Normal;
                }
                this.Activate();
                break;
            case "2":
                this.WindowState = FormWindowState.Minimized;
                break;
            case "3":
                IEAccessible ie2 = new IEAccessible();
                SHDocVw.ShellWindows shellWindows2 = new SHDocVw.ShellWindows();
                foreach (SHDocVw.InternetExplorer window in shellWindows2)
                {
                    ie2.Focus((IntPtr)window.HWND, getSite(0));
                }
                break;
            case "4":
                IEAccessible ie1 = new IEAccessible();
                SHDocVw.ShellWindows shellWindows1 = new SHDocVw.ShellWindows();
                foreach (SHDocVw.InternetExplorer window in shellWindows1)
                {
                    ie1.Focus((IntPtr)window.HWND, getSite(1));
                }
                break;
            case "5":
                IEAccessible ie3 = new IEAccessible();
                SHDocVw.ShellWindows shellWindows3 = new SHDocVw.ShellWindows();
                foreach (SHDocVw.InternetExplorer window in shellWindows3)
                {
                    ie3.Focus((IntPtr)window.HWND, getSite(2));
                }
                break;
            case "6":
                // --
                Process[] processes = Process.GetProcessesByName("iexplore");
                foreach (var item in processes)
                {
                    //ShowWindow(item.MainWindowHandle, SW_HIDE);
                    SetForegroundWindow(item.MainWindowHandle);
                }
                // --
                break;
        }
    }

    /// <summary>
    /// Get Pressed Key
```

```
/// </summary>
/// <param name="LParam"></param>
/// <returns></returns>
private Keys GetKey(IntPtr LParam)
{
   return (Keys)((LParam.ToInt32()) >> 16);
}

/// <summary>
/// Handle Key Event
/// </summary>
/// <param name="m"></param>
protected override void WndProc(ref Message m)
{
   if (m.Msg == Constants.WM_HOTKEY_MSG_ID)
   {
      switch (GetKey(m.LParam))
      {
         case Keys.D1:
            // the hotkey key is (Shift + 1)
            HandleHotkey("1");
            break;
         case Keys.D2:
            // the hotkey key is (Shift + 2)
            HandleHotkey("2");
            break;
         case Keys.D3:
            // the hotkey key is (Shift + 3)
            HandleHotkey("6");
            break;
         case Keys.A:
            // the hotkey key is (Shift + a)
            HandleHotkey("3");
            break;
         case Keys.S:
            // the hotkey key is (Shift + i)
            HandleHotkey("4");
            break;
         case Keys.D:
            // the hotkey key is (Shift + k)
            HandleHotkey("5");
            break;
      }
   }

   base.WndProc(ref m);
}

/// <summary>
/// Form Load Event
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void Form1_Load(object sender, EventArgs e)
{
   ghk1.Register();
   ghk2.Register();
   ghk3.Register();
   ghk4.Register();
   ghk5.Register();
   ghk6.Register();
}
```

```
/// <summary>
/// Form Close Event
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
   ghk1.Unregiser();
   ghk2.Unregiser();
   ghk3.Unregiser();
   ghk4.Unregiser();
   ghk5.Unregiser();
   ghk6.Unregiser();
}

#endregion

#region NAudio
/// <summary>
/// Start recording from MIC
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void button2_Click(object sender, EventArgs e)
{
   pictureBox2.BackColor = Color.Red;
   if (!debug)
   {
      // stop recognize before recode and filter
      speechRecognitionEngine.RecognizeAsyncCancel();
      speechRecognitionEngine.SetInputToNull();
   }

   // wait
   try
   {
      Thread.Sleep(10);
   }
   catch (Exception)
   {

   }

   waveSource = new WaveIn();
   waveSource.DataAvailable += new EventHandler<WaveInEventArgs>(waveIn_DataAvailable);
   waveSource.RecordingStopped += new EventHandler(waveSource_RecordingStopped);
   filter = BiQuadFilter.LowPassFilter(waveSource.WaveFormat.SampleRate, 10000, 1);
   waveFile = new WaveFileWriter(@"Test0001.wav", waveSource.WaveFormat);
   waveSource.StartRecording();
}

/// <summary>
/// Filter available data
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
void waveIn_DataAvailable(object sender, WaveInEventArgs e)
{
   short g = 0;
   var floatBuffer = new List<float>();
   for (int index = 0; index < e.BytesRecorded; index += 2)
   {
      short sample = BitConverter.ToInt16(e.Buffer, index);
```

```
            /**
             * Gain
             */
            float sample32 = (float)sample + g;
            sample32 /= (float)Int16.MaxValue;
            floatBuffer.Add(sample32);
        }

        /**
         * low pass filtering
         */
        for (int i = 0; i < floatBuffer.Count; i++)
        {
            waveFile.WriteSample(filter.Transform(floatBuffer[i]));
            waveFile.Flush();
        }
    }

    /// <summary>
    /// Recode stop event
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    void waveSource_RecordingStopped(object sender, EventArgs e)
    {

        if (waveSource != null)
        {
            waveSource.Dispose();
            waveSource = null;
        }

        if (waveFile != null)
        {
            waveFile.Dispose();
            waveFile = null;
        }

        if (!debug)
        {
            recognize();
        }
    }

    /// <summary>
    /// Stop recording from MIC
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void button3_Click(object sender, EventArgs e)
    {
        pictureBox2.BackColor = Color.Green;
        if (waveSource != null)
        {
            waveSource.StopRecording();
        }
        else
        {
            if (!debug)
            {
                recognize();
            }
```

```
        }
    }

    /// <summary>
    /// Start recognize
    /// </summary>
    private void recognize()
    {
        speechRecognitionEngine.SetInputToWaveFile("Test0001.wav");
        speechRecognitionEngine.RecognizeAsync(RecognizeMode.Multiple);
    }

    #endregion

    #region clustering
    /// <summary>
    /// Init data for clustering
    /// </summary>
    private void Populatecluster()
    {
        Dictionary<String, Dictionary<String, String[]>> dic1 = new Dictionary<string, Dictionary<String,
String[]>>();
        try
        {
            string[] lines = File.ReadAllLines(Environment.CurrentDirectory + "\\Cluster.txt");
            foreach (string line in lines)
            {
                // split the line
                var parts = line.Split(new char[] { ':' });

                if (dic1.ContainsKey(parts[0]))
                {
                    Dictionary<String, String[]> tdic = dic1[parts[0]];
                    tdic[parts[1]] = parts[2].Split(new char[] { ',' });
                }
                else
                {
                    Dictionary<String, String[]> tdic = new Dictionary<string, string[]>();
                    tdic[parts[1]] = parts[2].Split(new char[] { ',' });
                    dic1[parts[0]] = tdic;
                }
            }
        }
        catch (Exception ex)
        {
            throw ex;
        }

        /// teach BayesClassifier
        foreach (KeyValuePair<String, Dictionary<String, String[]>> entry in dic1)
        {
            BayesClassifier.Classifier m_Classifier1 = new BayesClassifier.Classifier();
            vm_Classifier.Add(m_Classifier1);
            Teach(entry.Value, m_Classifier1);
        }

    }

    /// <summary>
    /// Training data
    /// </summary>
    /// <param name="tdata"></param>
    private void Teach(Dictionary<String, String[]> tdata)
```

```
{
   foreach (KeyValuePair<String, String[]> entry in tdata)
   {
      m_Classifier.TeachPhrases(entry.Key, entry.Value);
   }
}

/// <summary>
/// Training data
/// </summary>
/// <param name="tdata"></param>
/// <param name="m_Classifier1"></param>
private void Teach(Dictionary<String, String[]> tdata, BayesClassifier.Classifier m_Classifier1)
{
   foreach (KeyValuePair<String, String[]> entry in tdata)
   {
      m_Classifier1.TeachPhrases(entry.Key, entry.Value);
   }
}

/// <summary>
/// Classify into cluster
/// </summary>
/// <param name="data"></param>
/// <returns></returns>
private string Classify(String data)
{

   /*
    * Example data => "FTTH EMAIL"
    */

   // convert string to stream
   byte[] byteArray = Encoding.UTF8.GetBytes(data);
   MemoryStream stream = new MemoryStream(byteArray);
   // convert stream to string
   StreamReader reader = new StreamReader(stream);
   Dictionary<string, double> score = m_Classifier.Classify(reader);
   ArrayList myAL = new ArrayList();

   foreach (string c in score.Keys)
   {
      string key = c;
      double val = score[c];
      myAL.Add(val);
   }

   myAL.Sort();
   double max = (double)myAL[myAL.Count-1];
   ArrayList fi = new ArrayList();

   foreach (string c in score.Keys)
   {
      string key = c;
      double val = score[c];
      if (max == val)
         fi.Add(key);
   }

   return string.Join("|", fi.ToArray());
}

/// <summary>
```

```
        /// Classify into cluster
        /// </summary>
        /// <param name="data"></param>
        /// <param name="m_Classifier1"></param>
        /// <returns></returns>
        private string Classify(String data, BayesClassifier.Classifier m_Classifier1)
        {

            /*
             * Example data => "FTTH EMAIL"
             */

            // convert string to stream
            byte[] byteArray = Encoding.UTF8.GetBytes(data);
            MemoryStream stream = new MemoryStream(byteArray);
            // convert stream to string
            StreamReader reader = new StreamReader(stream);
            Dictionary<string, double> score = m_Classifier1.Classify(reader);
            ArrayList myAL = new ArrayList();

            foreach (string c in score.Keys)
            {
                string key = c;
                double val = score[c];
                myAL.Add(val);
            }

            myAL.Sort();
            double max = (double)myAL[myAL.Count - 1];
            ArrayList fi = new ArrayList();

            foreach (string c in score.Keys)
            {
                string key = c;
                double val = score[c];
                if (max == val)
                    fi.Add(key);
            }

            return string.Join("|", fi.ToArray());
        }

        #endregion

        #region spellChecker
        /// <summary>
        /// capture words like "test test"
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="args"></param>
        private void SpellChecker_DoubledWord(object sender, NetSpell.SpellChecker.SpellingEventArgs
args)
        {
            // update text
            string s = spellChecker.Text;
        }

        /// <summary>
        /// capture words like "test"
        /// this word is correct
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="args"></param>
```

```
        private void SpellChecker_EndOfText(object sender, System.EventArgs args)
        {
            // update text
            string s = spellChecker.Text;
            textBox1.Text = textBox1.Text + "," + s;
        }

        /// <summary>
        /// capture words like "tes"
        /// this word is incorrect
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="args"></param>
        private void SpellChecker_MisspelledWord(object sender, NetSpell.SpellChecker.SpellingEventArgs
args)
        {
            // update text
            string s = spellChecker.Text;
        }

        /// <summary>
        /// check spellings of the words
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void textBox1_TextChanged(object sender, EventArgs e)
        {
            string temp = textBox1.Text.Trim();
            string word = (temp.Contains(",")) ? temp.Split(',').Last() : temp;
            if (word != "")
            {
                spellChecker.Text = word;
            }

            if (!temp.Contains(","))
            {
                prevrecognized = temp;
            }
            else
            {
                prevrecognized = temp.Split(',')[temp.Split(',').Length - 1];
            }
        }


        /// <summary>
        /// check spellings
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void button5_Click(object sender, EventArgs e)
        {
            spellChecker.SpellCheck();
        }

        #endregion

        #region Sites

        /// <summary>
        /// load sites for hot keys
        /// </summary>
        private void loadSites()
```

```
{
   try
   {
      string[] lines = File.ReadAllLines(Environment.CurrentDirectory + "\\SITES.txt");
      foreach (string line in lines)
      {
         sites.Add(line);
      }
   }
   catch (Exception ex)
   {
      throw ex;
   }
}

/// <summary>
/// get sites by index
/// </summary>
/// <param name="i"></param>
/// <returns></returns>
private string getSite(int i)
{
   try
   {
      string value = sites[i] as string;
      return value;
   }
   catch (Exception e)
   {
      return "not avail";
   }
}

#endregion

#region Markov

/// <summary>
/// Returns the number of steps required to transform the source string
/// into the target string.
/// </summary>
public static int ComputeLevenshteinDistance(string source, string target)
{
   if (string.IsNullOrEmpty(source))
      return string.IsNullOrEmpty(target) ? 0 : target.Length;

   if (string.IsNullOrEmpty(target))
      return string.IsNullOrEmpty(source) ? 0 : source.Length;

   int sourceLength = source.Length;
   int targetLength = target.Length;
   int[,] distance = new int[sourceLength + 1, targetLength + 1];

   // Step 1
   for (int i = 0; i <= sourceLength; distance[i, 0] = i++) ;
   for (int j = 0; j <= targetLength; distance[0, j] = j++) ;
   for (int i = 1; i <= sourceLength; i++)
   {
      for (int j = 1; j <= targetLength; j++)
      {
         // Step 2
         int cost = (target[j - 1] == source[i - 1]) ? 0 : 1;
         // Step 3
```

```
                distance[i, j] = Math.Min(Math.Min(distance[i - 1, j] + 1, distance[i, j - 1] + 1), distance[i - 1, j - 1] + cost);
            }
        }

        return distance[sourceLength, targetLength];
    }


    /// <summary>
    /// Calculate percentage similarity of two strings
    /// <param name="source">Source String to Compare with</param>
    /// <param name="target">Targeted String to Compare</param>
    /// <returns>Return Similarity between two strings from 0 to 1.0</returns>
    /// </summary>
    public static double CalculateSimilarity(string source, string target)
    {
        if (string.IsNullOrEmpty(source))
            return string.IsNullOrEmpty(target) ? 1 : 0;


        if (string.IsNullOrEmpty(target))
            return string.IsNullOrEmpty(source) ? 1 : 0;


        double stepsToSame = ComputeLevenshteinDistance(source, target);
        return (1.0 - (stepsToSame / (double)Math.Max(source.Length, target.Length)));
    }


    /// <summary>
    /// tokenise a string into words (regex definition of word)
    /// </summary>
    /// <param name="subject"></param>
    /// <returns></returns>
    private static IEnumerable<string> Split(string subject)
    {
        List<string> tokens = new List<string>();
        Regex regex = new Regex(@"(\W+)");
        string[] arr = regex.Split(subject);
        for (int i = 0; i < arr.Length; i++)
        {
            if (arr[i].Equals(" "))
                continue;

            if (arr[i].Equals(""))
                continue;

            if (arr[i].Contains("\n"))
                continue;

            if (arr[i].Contains("."))
                continue;

            tokens.Add(arr[i]);
        }

        return tokens;
    }


    /// <summary>
    /// Preprocess string
```

```
        /// </summary>
        /// <param name="p"></param>
        /// <returns></returns>
        private static string Tidy(string p)
        {
            string result = p.Replace('\t', ' ');
            string compress = result;

            do
            {
                result = compress;
                compress = result.Replace("  ", " ");
            }
            while (result != compress);
            return result;
        }

        /// <summary>
        /// load Markov
        /// </summary>
        private void LoadMarkov()
        {
            try
            {
                string seed = "";

                string[] lines = File.ReadAllLines(Environment.CurrentDirectory + "\\Markov.txt");
                foreach (string line in lines)
                {
                    seed = line;
                    break;
                }

                seed = Tidy(seed);

                // tokenise the input string
                var seedList = new List<string>(Split(seed.ToLower()));
                // create a chain with a window size of 4
                chain = new Chain<string>(seedList, 4);

                string[] startw = File.ReadAllLines(Environment.CurrentDirectory + "\\MarkovStartW.txt");

                for (int i = 0; i < startw.Length; i++)
                {
                    chainstart.Add(startw[i]);
                }

            }
            catch (Exception ex)
            {
                throw ex;
            }
        }

        #endregion

    }
}
```

Appendix B

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace MSC
{
   /// <summary>
   /// BiQuad filter
   /// </summary>
   public class BiQuadFilter
   {
      // coefficients
      private double a0;
      private double a1;
      private double a2;
      private double a3;
      private double a4;

      // state
      private float x1;
      private float x2;
      private float y1;
      private float y2;

      /// <summary>
      /// Passes a single sample through the filter
      /// </summary>
      /// <param name="inSample">Input sample</param>
      /// <returns>Output sample</returns>
      public float Transform(float inSample)
      {
         // compute result
         var result = a0 * inSample + a1 * x1 + a2 * x2 - a3 * y1 - a4 * y2;

         // shift x1 to x2, sample to x1
         x2 = x1;
         x1 = inSample;

         // shift y1 to y2, result to y1
         y2 = y1;
         y1 = (float)result;

         return y1;
         //return inSample;
      }

      private void SetCoefficients(double aa0, double aa1, double aa2, double b0, double b1, double b2)
      {
         // precompute the coefficients
         a0 = b0 / aa0;
         a1 = b1 / aa0;
         a2 = b2 / aa0;
         a3 = aa1 / aa0;
         a4 = aa2 / aa0;
      }

      /// <summary>
      /// Set this up as a low pass filter
      /// </summary>
```

```
/// <param name="sampleRate">Sample Rate</param>
/// <param name="cutoffFrequency">Cut-off Frequency</param>
/// <param name="q">Bandwidth</param>
public void SetLowPassFilter(float sampleRate, float cutoffFrequency, float q)
{
    // H(s) = 1 / (s^2 + s/Q + 1)
    var w0 = 2 * Math.PI * cutoffFrequency / sampleRate;
    var cosw0 = Math.Cos(w0);
    var alpha = Math.Sin(w0) / (2 * q);

    var b0 = (1 - cosw0) / 2;
    var b1 = 1 - cosw0;
    var b2 = (1 - cosw0) / 2;
    var aa0 = 1 + alpha;
    var aa1 = -2 * cosw0;
    var aa2 = 1 - alpha;
    SetCoefficients(aa0, aa1, aa2, b0, b1, b2);
}

/// <summary>
/// Set this up as a peaking EQ
/// </summary>
/// <param name="sampleRate">Sample Rate</param>
/// <param name="centreFrequency">Centre Frequency</param>
/// <param name="q">Bandwidth (Q)</param>
/// <param name="dbGain">Gain in decibels</param>
public void SetPeakingEq(float sampleRate, float centreFrequency, float q, float dbGain)
{
    // H(s) = (s^2 + s*(A/Q) + 1) / (s^2 + s/(A*Q) + 1)
    var w0 = 2 * Math.PI * centreFrequency / sampleRate;
    var cosw0 = Math.Cos(w0);
    var sinw0 = Math.Sin(w0);
    var alpha = sinw0 / (2 * q);
    var a = Math.Pow(10, dbGain / 40);     // TODO: should we square root this value?

    var b0 = 1 + alpha * a;
    var b1 = -2 * cosw0;
    var b2 = 1 - alpha * a;
    var aa0 = 1 + alpha / a;
    var aa1 = -2 * cosw0;
    var aa2 = 1 - alpha / a;
    SetCoefficients(aa0, aa1, aa2, b0, b1, b2);
}

/// <summary>
/// Set this as a high pass filter
/// </summary>
public void SetHighPassFilter(float sampleRate, float cutoffFrequency, float q)
{
    // H(s) = s^2 / (s^2 + s/Q + 1)
    var w0 = 2 * Math.PI * cutoffFrequency / sampleRate;
    var cosw0 = Math.Cos(w0);
    var alpha = Math.Sin(w0) / (2 * q);

    var b0 = (1 + cosw0) / 2;
    var b1 = -(1 + cosw0);
    var b2 = (1 + cosw0) / 2;
    var aa0 = 1 + alpha;
    var aa1 = -2 * cosw0;
    var aa2 = 1 - alpha;
    SetCoefficients(aa0, aa1, aa2, b0, b1, b2);
}
```

```
        /// <summary>
        /// Create a low pass filter
        /// </summary>
        public static BiQuadFilter LowPassFilter(float sampleRate, float cutoffFrequency, float q)
        {
            var filter = new BiQuadFilter();
            filter.SetLowPassFilter(sampleRate, cutoffFrequency, q);
            return filter;
        }

        /// <summary>
        /// Create a High pass filter
        /// </summary>
        public static BiQuadFilter HighPassFilter(float sampleRate, float cutoffFrequency, float q)
        {
            var filter = new BiQuadFilter();
            filter.SetHighPassFilter(sampleRate, cutoffFrequency, q);
            return filter;
        }

        /// <summary>
        /// Create a bandpass filter with constant skirt gain
        /// </summary>
        public static BiQuadFilter BandPassFilterConstantSkirtGain(float sampleRate, float centreFrequency,
float q)
        {
            // H(s) = s / (s^2 + s/Q + 1)  (constant skirt gain, peak gain = Q)
            var w0 = 2 * Math.PI * centreFrequency / sampleRate;
            var cosw0 = Math.Cos(w0);
            var sinw0 = Math.Sin(w0);
            var alpha = sinw0 / (2 * q);

            var b0 = sinw0 / 2; // =   Q*alpha
            var b1 = 0;
            var b2 = -sinw0 / 2; // =  -Q*alpha
            var a0 = 1 + alpha;
            var a1 = -2 * cosw0;
            var a2 = 1 - alpha;
            return new BiQuadFilter(a0, a1, a2, b0, b1, b2);
        }

        /// <summary>
        /// Create a bandpass filter with constant peak gain
        /// </summary>
        public static BiQuadFilter BandPassFilterConstantPeakGain(float sampleRate, float centreFrequency,
float q)
        {
            // H(s) = (s/Q) / (s^2 + s/Q + 1)      (constant 0 dB peak gain)
            var w0 = 2 * Math.PI * centreFrequency / sampleRate;
            var cosw0 = Math.Cos(w0);
            var sinw0 = Math.Sin(w0);
            var alpha = sinw0 / (2 * q);

            var b0 = alpha;
            var b1 = 0;
            var b2 = -alpha;
            var a0 = 1 + alpha;
            var a1 = -2 * cosw0;
            var a2 = 1 - alpha;
            return new BiQuadFilter(a0, a1, a2, b0, b1, b2);
        }

        /// <summary>
```

```
        /// Creates a notch filter
        /// </summary>
        public static BiQuadFilter NotchFilter(float sampleRate, float centreFrequency, float q)
        {
            // H(s) = (s^2 + 1) / (s^2 + s/Q + 1)
            var w0 = 2 * Math.PI * centreFrequency / sampleRate;
            var cosw0 = Math.Cos(w0);
            var sinw0 = Math.Sin(w0);
            var alpha = sinw0 / (2 * q);

            var b0 = 1;
            var b1 = -2 * cosw0;
            var b2 = 1;
            var a0 = 1 + alpha;
            var a1 = -2 * cosw0;
            var a2 = 1 - alpha;
            return new BiQuadFilter(a0, a1, a2, b0, b1, b2);
        }

        /// <summary>
        /// Creaes an all pass filter
        /// </summary>
        public static BiQuadFilter AllPassFilter(float sampleRate, float centreFrequency, float q)
        {
            //H(s) = (s^2 - s/Q + 1) / (s^2 + s/Q + 1)
            var w0 = 2 * Math.PI * centreFrequency / sampleRate;
            var cosw0 = Math.Cos(w0);
            var sinw0 = Math.Sin(w0);
            var alpha = sinw0 / (2 * q);

            var b0 = 1 - alpha;
            var b1 = -2 * cosw0;
            var b2 = 1 + alpha;
            var a0 = 1 + alpha;
            var a1 = -2 * cosw0;
            var a2 = 1 - alpha;
            return new BiQuadFilter(a0, a1, a2, b0, b1, b2);
        }

        /// <summary>
        /// Create a Peaking EQ
        /// </summary>
        public static BiQuadFilter PeakingEQ(float sampleRate, float centreFrequency, float q, float dbGain)
        {
            var filter = new BiQuadFilter();
            filter.SetPeakingEq(sampleRate, centreFrequency, q, dbGain);
            return filter;
        }

        /// <summary>
        /// H(s) = A * (s^2 + (sqrt(A)/Q)*s + A)/(A*s^2 + (sqrt(A)/Q)*s + 1)
        /// </summary>
        /// <param name="sampleRate"></param>
        /// <param name="cutoffFrequency"></param>
        /// <param name="shelfSlope">a "shelf slope" parameter (for shelving EQ only).
        /// When S = 1, the shelf slope is as steep as it can be and remain monotonically
        /// increasing or decreasing gain with frequency.  The shelf slope, in dB/octave,
        /// remains proportional to S for all other values for a fixed f0/Fs and dBgain.</param>
        /// <param name="dbGain">Gain in decibels</param>
        public static BiQuadFilter LowShelf(float sampleRate, float cutoffFrequency, float shelfSlope, float
dbGain)
        {
            var w0 = 2 * Math.PI * cutoffFrequency / sampleRate;
```

```
            var cosw0 = Math.Cos(w0);
            var sinw0 = Math.Sin(w0);
            var a = Math.Pow(10, dbGain / 40);     // TODO: should we square root this value?
            var alpha = sinw0 / 2 * Math.Sqrt((a + 1 / a) * (1 / shelfSlope - 1) + 2);
            var temp = 2 * Math.Sqrt(a) * alpha;

            var b0 = a * ((a + 1) - (a - 1) * cosw0 + temp);
            var b1 = 2 * a * ((a - 1) - (a + 1) * cosw0);
            var b2 = a * ((a + 1) - (a - 1) * cosw0 - temp);
            var a0 = (a + 1) + (a - 1) * cosw0 + temp;
            var a1 = -2 * ((a - 1) + (a + 1) * cosw0);
            var a2 = (a + 1) + (a - 1) * cosw0 - temp;
            return new BiQuadFilter(a0, a1, a2, b0, b1, b2);
        }

        /// <summary>
        /// H(s) = A * (A*s^2 + (sqrt(A)/Q)*s + 1)/(s^2 + (sqrt(A)/Q)*s + A)
        /// </summary>
        /// <param name="sampleRate"></param>
        /// <param name="cutoffFrequency"></param>
        /// <param name="shelfSlope"></param>
        /// <param name="dbGain"></param>
        /// <returns></returns>
        public static BiQuadFilter HighShelf(float sampleRate, float cutoffFrequency, float shelfSlope, float
dbGain)
        {
            var w0 = 2 * Math.PI * cutoffFrequency / sampleRate;
            var cosw0 = Math.Cos(w0);
            var sinw0 = Math.Sin(w0);
            var a = Math.Pow(10, dbGain / 40);     // TODO: should we square root this value?
            var alpha = sinw0 / 2 * Math.Sqrt((a + 1 / a) * (1 / shelfSlope - 1) + 2);
            var temp = 2 * Math.Sqrt(a) * alpha;

            var b0 = a * ((a + 1) + (a - 1) * cosw0 + temp);
            var b1 = -2 * a * ((a - 1) + (a + 1) * cosw0);
            var b2 = a * ((a + 1) + (a - 1) * cosw0 - temp);
            var a0 = (a + 1) - (a - 1) * cosw0 + temp;
            var a1 = 2 * ((a - 1) - (a + 1) * cosw0);
            var a2 = (a + 1) - (a - 1) * cosw0 - temp;
            return new BiQuadFilter(a0, a1, a2, b0, b1, b2);
        }

        private BiQuadFilter()
        {
            // zero initial samples
            x1 = x2 = 0;
            y1 = y2 = 0;
        }

        private BiQuadFilter(double a0, double a1, double a2, double b0, double b1, double b2)
        {
            SetCoefficients(a0, a1, a2, b0, b1, b2);

            // zero initial samples
            x1 = x2 = 0;
            y1 = y2 = 0;
        }
    }
}
```

Appendix C

```
using System;
using System.Collections.Generic;
using System.Text;

namespace BayesClassifier
{
	/// <summary>
	/// Naive Bayesian classifier</summary>
	/// <remarks>
	/// It suppports exclusion of words but not Phrases
	/// </remarks>
	public class Classifier : BayesClassifier.IClassifier
	{
		SortedDictionary<string, ICategory> m_Categories;
		ExcludedWords m_ExcludedWords;

		public Classifier()
		{
			m_Categories = new SortedDictionary<string, ICategory>();
			m_ExcludedWords = new ExcludedWords();
			m_ExcludedWords.InitDefault();
		}

		/// <summary>
		/// Gets total number of word occurences over all categories</summary>
		int CountTotalWordsInCategories()
		{
			int total = 0;
			foreach (Category cat in m_Categories.Values)
			{
				total += cat.TotalWords;
			}
			return total;
		}

		/// <summary>
		/// Gets or creates a category</summary>
		ICategory GetOrCreateCategory(string cat)
		{
			ICategory c;
			if (!m_Categories.TryGetValue(cat, out c))
			{
				c = new Category(cat, m_ExcludedWords);
				m_Categories.Add(cat, c);
			}
			return c;
		}

		/// <summary>
		/// Trains this Category from a word or phrase<\summary>
		public void TeachPhrases(string cat, string[] phrases)
		{
			GetOrCreateCategory(cat).TeachPhrases(phrases);
		}

		/// <summary>
		/// Trains this Category from a word or phrase<\summary>
		public void TeachCategory(string cat, System.IO.TextReader tr)
		{
			GetOrCreateCategory(cat).TeachCategory(tr);
```

```
                }

                /// <summary>
                /// Classifies a text<\summary>
                /// <returns>
                /// returns classification values for the text, the higher, the better is the match.</returns>
                public Dictionary<string, double> Classify(System.IO.StreamReader tr)
                {
                        Dictionary<string, double> score = new Dictionary<string, double>();
                        foreach (KeyValuePair<string, ICategory> cat in m_Categories)
                        {
                                score.Add(cat.Value.Name, 0.0);
                        }

                        EnumerableCategory words_in_file = new EnumerableCategory("",
m_ExcludedWords);
                        words_in_file.TeachCategory(tr);

                        foreach (KeyValuePair<string, PhraseCount> kvp1 in words_in_file)
                        {
                                PhraseCount pc_in_file = kvp1.Value;
                                foreach (KeyValuePair<string, ICategory> kvp in m_Categories)
                                {
                                ICategory cat = kvp.Value;
                                int count = cat.GetPhraseCount(pc_in_file.RawPhrase);
                                if (0 < count)
                                {
                                score[cat.Name] += System.Math.Log((double)count /
(double)cat.TotalWords);
                                }
                                else
                                {
                                score[cat.Name] += System.Math.Log(0.01 / (double)cat.TotalWords);
                                }
                                System.Diagnostics.Trace.WriteLine(pc_in_file.RawPhrase.ToString()
+ "(" +
                                                cat.Name + ")" + score[cat.Name]);
                                }
                        }
                        foreach (KeyValuePair<string, ICategory> kvp in m_Categories)
                        {
                        ICategory cat = kvp.Value;
                                score[cat.Name] += System.Math.Log((double)cat.TotalWords /
                        (double)this.CountTotalWordsInCategories());
                        }
                        return score;
                }
        }
}
```

```
using System;
using System.Collections.Generic;

namespace BayesClassifier
{
   #region Interfaces
        /// <summary>
        /// Classifier methods.
        /// </summary>
        public interface IClassifier
        {
                /// <summary>
                /// Trains this Category from a word or phrase<\summary>
                void TeachPhrases(string cat, string[] phrases);

                /// <summary>
                /// Trains this Category from a word or phrase<\summary>
                void TeachCategory(string cat, System.IO.TextReader tr);

                /// <summary>
                /// Classifies a text<\summary>
                /// <returns>
                /// returns classification values for the text, the higher, the better is the match.</returns>
                Dictionary<string, double> Classify(System.IO.StreamReader tr);
        }

        /// <summary>
        /// Category methods.
        /// </summary>
        interface ICategory
        {
                string Name { get; }
                /// <summary>
                /// Reset all trained data<\summary>
                void Reset();
                /// <summary>
                /// Gets a PhraseCount for Phrase or 0 if phrase not present<\summary>
                int GetPhraseCount(string phrase);

                /// <summary>
                /// Trains this Category from a file<\summary>
                void TeachCategory(System.IO.TextReader reader);
                /// <summary>
                /// Trains this Category from a word or phrase<\summary>
                void TeachPhrase(string rawPhrase);
                /// <summary>
                /// Trains this Category from a word or phrase array<\summary>
                void TeachPhrases(string[] words);
                /// <value>
                /// Gets total number of word occurences in this category</value>
                int TotalWords { get; }
        }
   #endregion
}
```

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Text.RegularExpressions;

namespace BayesClassifier
{
	/// <summary>
	/// stores occurence counter for words or phrases</summary>
	class PhraseCount
	{
		string m_RawPhrase;

		public PhraseCount(string rawPhrase)
		{
			m_RawPhrase = rawPhrase;
			m_Count = 0;
		}

		/// <value>
		/// Stores the raw Phrase, the real matching phrase is stored as key to this element</value>
		/// <seealso cref="DePhrase(string)">
		/// See DePhrase </seealso>
		public string RawPhrase
		{
			get { return m_RawPhrase; }
			//set { m_RawPhrase = value; }
		}

		int m_Count;

		/// <value>
		/// Count Accessor</value>
		public int Count
		{
			get { return m_Count; }
			set { m_Count = value; }
		}
	}
}
```

```
using System;
using System.Collections.Generic;
using System.Collections;
using System.Text;
using System.Text.RegularExpressions;
namespace BayesClassifier
{
	/// <summary>
	/// Represents a Enumerable Bayesian category - that is contains a list of phrases with their
occurence counts <\summary>
	class EnumerableCategory : Category, IEnumerable<KeyValuePair<string, PhraseCount>>
	{
		public EnumerableCategory(string Cat, ExcludedWords Excluded) : base(Cat, Excluded)
		{
		}

		IEnumerator IEnumerable.GetEnumerator()
		{
			return GetEnumerator();
		}
		public IEnumerator<KeyValuePair<string, PhraseCount>> GetEnumerator()
		{
			return m_Phrases.GetEnumerator();
		}

	}


	/// <summary>
	/// Represents a Bayesian category - that is contains a list of phrases with their occurence counts
<\summary>
	class Category : BayesClassifier.ICategory
	{
		protected System.Collections.Generic.SortedDictionary<string, PhraseCount> m_Phrases;
		int m_TotalWords;
		string m_Name;
		ExcludedWords m_Excluded;

		public string Name
		{
			get { return m_Name; }
			//set { m_Name = value; }
		}
		/// <value>
		/// Gets total number of word occurences in this category</value>
		public int TotalWords
		{
			get { return m_TotalWords; }
			//set { m_TotalWords = value; }
		}

		public Category(string cat, ExcludedWords excluded)
		{
			m_Phrases = new SortedDictionary<string, PhraseCount>();
			m_Excluded = excluded;
			m_Name = cat;
		}

		/// <summary>
		/// Gets a Count for Phrase or 0 if not present<\summary>
		public int GetPhraseCount(string phrase)
		{
			PhraseCount pc;
```

```
		if (m_Phrases.TryGetValue(phrase, out pc))
			return pc.Count;
		else
			return 0;
	}

	/// <summary>
	/// Reset all trained data<\summary>
	public void Reset()
	{
		m_TotalWords = 0;
		m_Phrases.Clear();
	}

	System.Collections.Generic.SortedDictionary<string, PhraseCount> Phrases
	{
		get { return m_Phrases; }
		//set { m_Phrases = value; }
	}

	/// <summary>
	/// Trains this Category from a file<\summary>
	public void TeachCategory(System.IO.TextReader reader)
	{
		//System.Diagnostics.Debug.Assert(line.Length < 512);
		Regex re = new Regex(@"(\w+)\W*", RegexOptions.Compiled);
		string line;
		while (null != (line = reader.ReadLine()))
		{
			Match m = re.Match(line);
			while (m.Success)
			{
				string word = m.Groups[1].Value;
				TeachPhrase(word);
				m = m.NextMatch();
			}
		}
	}

	/// <summary>
	/// Trains this Category from a word or phrase array<\summary>
	/// <seealso cref="DePhrase(string)">
	/// See DePhrase </seealso>
	public void TeachPhrases(string[] words)
	{
		foreach (string word in words)
		{
			TeachPhrase(word);
		}
	}

	/// <summary>
	/// Trains this Category from a word or phrase<\summary>
	/// <seealso cref="DePhrase(string)">
	/// See DePhrase </seealso>
	public void TeachPhrase(string rawPhrase)
	{
		if ((null != m_Excluded) && (m_Excluded.IsExcluded(rawPhrase)))
			return;

		PhraseCount pc;
		string Phrase = DePhrase(rawPhrase);
		if (!m_Phrases.TryGetValue(Phrase, out pc))
```

```
                {
                        pc = new PhraseCount(rawPhrase);
                        m_Phrases.Add(Phrase, pc);
                }
                pc.Count++;
                m_TotalWords++;
        }

        static Regex ms_PhraseRegEx = new Regex(@"\W", RegexOptions.Compiled);

        /// <summary>
        /// Checks if a string is a phrase (that is a string with whitespaces)<\summary>
        /// <returns>
        /// true or false</returns>
        /// <seealso cref="DePhrase(string)">
        /// See DePhrase </seealso>
        public static bool CheckIsPhrase(string s)
        {
                return ms_PhraseRegEx.IsMatch(s);
        }

        /// <summary>
        /// Trnasforms a string into a phrase (that is a string with whitespaces)<\summary>
        /// <returns>
        /// dephrased string</returns>
        /// <remarks>
        /// if something like "lone Rhino" is considered a sinlge Phrase, then our word matching
algorithm is
        /// is transforming it into a single Word "lone Rhino" -> "loneRhino"
        /// Currently this feature is not used!
        /// </remarks>
        public static string DePhrase(string s)
        {
                return ms_PhraseRegEx.Replace(s, @"");
        }

    }
}
```

Appendix D

```
using Accessibility;
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Runtime.InteropServices;
using System.Windows.Forms;

namespace ADDIProtocol
{
   public class IEAccessible
   {
      private enum OBJID : uint
      {
         OBJID_WINDOW = 0x00000000,
      }

      private const int IE_ACTIVE_TAB = 2097154;
      private const int CHILDID_SELF = 0;
      private IAccessible accessible;

      private IEAccessible[] Children
      {
         get
         {
            int num = 0;
            object[] res = GetAccessibleChildren(accessible, out num);

            if (res == null) return new IEAccessible[0];

            List<IEAccessible> list = new List<IEAccessible>(res.Length);

            foreach (object obj in res)
            {
               IAccessible acc = obj as IAccessible;

               if (acc != null) list.Add(new IEAccessible(acc));
            }

            return list.ToArray();
         }
      }

      private string Name
      {
         get
         {
            return accessible.get_accName(CHILDID_SELF);
         }
      }

      private int ChildCount
      {
         get
         {
            return accessible.accChildCount;
         }
      }

      public IEAccessible()
      { }
```

```
        public IEAccessible(IntPtr ieHandle, string tabCaptionToActivate)
        {
            int tabCount = GetTabCount(ieHandle);
            if (tabCount > 0)
            {
                AccessibleObjectFromWindow(GetDirectUIHWND(ieHandle), OBJID.OBJID_WINDOW, ref
accessible);

                if (accessible == null) throw new Exception();

                IEAccessible ieDirectUIHWND = new IEAccessible(ieHandle);

                foreach (IEAccessible accessor in ieDirectUIHWND.Children)
                {
                    foreach (IEAccessible child in accessor.Children)
                    {
                        foreach (IEAccessible tab in child.Children)
                        {
                            if (tab.Name.ToLower().Contains(tabCaptionToActivate.ToLower()))
                            {
                                tab.Activate();
                                return;
                            }
                        }
                    }
                }
            }
        }

        public bool Focus(IntPtr ieHandle, string tabCaptionToActivate)
        {

            int tabCount = GetTabCount(ieHandle);
            if (tabCount > 0)
            {
                AccessibleObjectFromWindow(GetDirectUIHWND(ieHandle), OBJID.OBJID_WINDOW, ref
accessible);
                if (accessible == null) throw new Exception();

                IEAccessible ieDirectUIHWND = new IEAccessible(ieHandle);
                foreach (IEAccessible accessor in ieDirectUIHWND.Children)
                {
                    foreach (IEAccessible child in accessor.Children)
                    {
                        foreach (IEAccessible tab in child.Children)
                        {
                            if (tab.Name.ToLower().Contains(tabCaptionToActivate.ToLower()))
                            {
                                tab.Activate();
                                return true;
                            }
                        }
                    }
                }
            }

            return false;
        }

        public bool Is(IntPtr ieHandle, string tabCaptionToActivate)
        {
            int tabCount = GetTabCount(ieHandle);
```

```
        if (tabCount > 0)
        {
            AccessibleObjectFromWindow(GetDirectUIHWND(ieHandle), OBJID.OBJID_WINDOW, ref
accessible);

            if (accessible == null) throw new Exception();

            IEAccessible ieDirectUIHWND = new IEAccessible(ieHandle);

            foreach (IEAccessible accessor in ieDirectUIHWND.Children)
            {
                foreach (IEAccessible child in accessor.Children)
                {
                    foreach (IEAccessible tab in child.Children)
                    {
                        if (tab.Name.ToLower().Contains(tabCaptionToActivate.ToLower()))
                        {
                            return true;
                        }
                    }
                }
            }
        }

        return false;
    }

    public IEAccessible(IntPtr ieHandle, int tabIndexToActivate)
    {
        AccessibleObjectFromWindow(GetDirectUIHWND(ieHandle), OBJID.OBJID_WINDOW, ref
accessible);

        if (accessible == null) throw new Exception();

        int index = 0;
        IEAccessible ieDirectUIHWND = new IEAccessible(ieHandle);

        foreach (IEAccessible accessor in ieDirectUIHWND.Children)
        {
            foreach (IEAccessible child in accessor.Children)
            {
                foreach (IEAccessible tab in child.Children)
                {
                    if (tabIndexToActivate >= child.ChildCount - 1) return;

                    if (index == tabIndexToActivate)
                    {
                        tab.Activate();

                        return;
                    }

                    index++;
                }
            }
        }
    }

    private IEAccessible(IntPtr ieHandle)
    {
        AccessibleObjectFromWindow(GetDirectUIHWND(ieHandle), OBJID.OBJID_WINDOW, ref
accessible);
```

```
            if (accessible == null) throw new Exception();
        }

        public string GetActiveTabUrl(IntPtr ieHandle)
        {
            AccessibleObjectFromWindow(GetDirectUIHWND(ieHandle), OBJID.OBJID_WINDOW, ref
accessible);

            if (accessible == null) throw new Exception();

            IEAccessible ieDirectUIHWND = new IEAccessible(ieHandle);

            foreach (IEAccessible accessor in ieDirectUIHWND.Children)
            {
                foreach (IEAccessible child in accessor.Children)
                {
                    foreach (IEAccessible tab in child.Children)
                    {
                        object tabIndex = tab.accessible.get_accState(CHILDID_SELF);

                        if ((int)tabIndex == IE_ACTIVE_TAB)
                        {
                            string description = tab.accessible.get_accDescription(CHILDID_SELF);

                            if (!string.IsNullOrEmpty(description))
                            {
                                if (description.Contains(Environment.NewLine))
                                {
                                    string url =
description.Substring(description.IndexOf(Environment.NewLine)).Trim();

                                    return url;
                                }
                            }
                        }
                    }
                }
            }

            return String.Empty;
        }

        public int GetActiveTabIndex(IntPtr ieHandle)
        {
            AccessibleObjectFromWindow(GetDirectUIHWND(ieHandle), OBJID.OBJID_WINDOW, ref
accessible);

            if (accessible == null) throw new Exception();

            int index = 0;
            IEAccessible ieDirectUIHWND = new IEAccessible(ieHandle);

            foreach (IEAccessible accessor in ieDirectUIHWND.Children)
            {
                foreach (IEAccessible child in accessor.Children)
                {
                    foreach (IEAccessible tab in child.Children)
                    {
                        object tabIndex = tab.accessible.get_accState(0);

                        if ((int)tabIndex == IE_ACTIVE_TAB) return index;

                        index++;
```

```
            }
        }
    }

    return -1;
}

public string GetActiveTabCaption(IntPtr ieHandle)
{
    AccessibleObjectFromWindow(GetDirectUIHWND(ieHandle), OBJID.OBJID_WINDOW, ref
accessible);

    if (accessible == null) throw new Exception();

    IEAccessible ieDirectUIHWND = new IEAccessible(ieHandle);

    foreach (IEAccessible accessor in ieDirectUIHWND.Children)
    {
        foreach (IEAccessible child in accessor.Children)
        {
            foreach (IEAccessible tab in child.Children)
            {
                object tabIndex = tab.accessible.get_accState(0);


                if ((int)tabIndex == IE_ACTIVE_TAB) return tab.Name;
            }
        }
    }

    return String.Empty;
}

public List<string> GetTabCaptions(IntPtr ieHandle)
{
    AccessibleObjectFromWindow(GetDirectUIHWND(ieHandle), OBJID.OBJID_WINDOW, ref
accessible);

    if (accessible == null) throw new Exception();

    IEAccessible ieDirectUIHWND = new IEAccessible(ieHandle);
    List<string> captionList = new List<string>();

    foreach (IEAccessible accessor in ieDirectUIHWND.Children)
    {
        foreach (IEAccessible child in accessor.Children)
            foreach (IEAccessible tab in child.Children) captionList.Add(tab.Name);
    }

    if (captionList.Count > 0) captionList.RemoveAt(captionList.Count - 1);

    return captionList;
}

public int GetTabCount(IntPtr ieHandle)
{
    AccessibleObjectFromWindow(GetDirectUIHWND(ieHandle), OBJID.OBJID_WINDOW, ref
accessible);

    if (accessible == null) return 0; // throw new Exception();

    IEAccessible ieDirectUIHWND = new IEAccessible(ieHandle);
```

```
    foreach (IEAccessible accessor in ieDirectUIHWND.Children)
    {
        foreach (IEAccessible child in accessor.Children)
        {
            foreach (IEAccessible tab in child.Children) return child.ChildCount - 1;
        }
    }

    return 0;
}

//private IntPtr GetDirectUIHWND(IntPtr ieFrame)
//{
//    IntPtr directUI = FindWindowEx(ieFrame, IntPtr.Zero, "CommandBarClass", null);
//    directUI = FindWindowEx(directUI, IntPtr.Zero, "ReBarWindow32", null);
//    directUI = FindWindowEx(directUI, IntPtr.Zero, "TabBandClass", null);
//    directUI = FindWindowEx(directUI, IntPtr.Zero, "DirectUIHWND", null);

//    return directUI;
//}

private IntPtr GetDirectUIHWND(IntPtr ieFrame)
{
    // For IE 8:
    var directUI = FindWindowEx(ieFrame, IntPtr.Zero, "CommandBarClass", null);
    directUI = FindWindowEx(directUI, IntPtr.Zero, "ReBarWindow32", null);
    directUI = FindWindowEx(directUI, IntPtr.Zero, "TabBandClass", null);
    directUI = FindWindowEx(directUI, IntPtr.Zero, "DirectUIHWND", null);

    if (directUI == IntPtr.Zero)
    {
        // For IE 9:
        directUI = FindWindowEx(ieFrame, IntPtr.Zero, "WorkerW", null);
        directUI = FindWindowEx(directUI, IntPtr.Zero, "ReBarWindow32", null);
        directUI = FindWindowEx(directUI, IntPtr.Zero, "TabBandClass", null);
        directUI = FindWindowEx(directUI, IntPtr.Zero, "DirectUIHWND", null);
    }

    return directUI;
}

private IEAccessible(IAccessible acc)
{
    if (acc == null) throw new Exception();

    accessible = acc;
}

private void Activate()
{
    accessible.accDoDefaultAction(CHILDID_SELF);
}

private static object[] GetAccessibleChildren(IAccessible ao, out int childs)
{
    childs = 0;
    object[] ret = null;
    int count = ao.accChildCount;

    if (count > 0)
    {
        ret = new object[count];
        AccessibleChildren(ao, 0, count, ret, out childs);
```

```
            }

            return ret;
        }

        #region Interop
        [DllImport("user32.dll", SetLastError = true)]
        private static extern IntPtr FindWindowEx(IntPtr hwndParent, IntPtr hwndChildAfter, string lpszClass,
string lpszWindow);

        private static int AccessibleObjectFromWindow(IntPtr hwnd, OBJID idObject, ref IAccessible acc)
        {
            object obj = null;
            Guid guid = new Guid("{618736e0-3c3d-11cf-810c-00aa00389b71}"); // IAccessibleobject obj =
null;
            int num = AccessibleObjectFromWindow(hwnd, (uint)idObject, ref guid, ref obj);

            acc = (IAccessible)obj;

            return num;
        }

        [DllImport("oleacc.dll")]
        private static extern int AccessibleObjectFromWindow(IntPtr hwnd, uint id, ref Guid iid, [In, Out,
MarshalAs(UnmanagedType.IUnknown)] ref object ppvObject);

        [DllImport("oleacc.dll")]
        private static extern int AccessibleChildren(IAccessible paccContainer, int iChildStart, int cChildren,
[In, Out, MarshalAs(UnmanagedType.LPArray, SizeParamIndex = 2)] object[] rgvarChildren, out int
pcObtained);
        #endregion
    }}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Runtime.InteropServices;

namespace MSC
{
    class GlobalHotkey
    {
        [DllImport("user32.dll")]
        private static extern bool RegisterHotKey(IntPtr hWnd, int id, int fsModifiers, int vk);

        [DllImport("user32.dll")]
        private static extern bool UnregisterHotKey(IntPtr hWnd, int id);

        private int modifier;
        private int key;
        private IntPtr hWnd;
        private int id;

        public GlobalHotkey(int modifier, Keys key, Form form)
        {
            this.modifier = modifier;
            this.key = (int)key;
            this.hWnd = form.Handle;
            id = this.GetHashCode();
        }

        public override int GetHashCode()
        {
            return modifier ^ key ^ hWnd.ToInt32();
        }

        public bool Register()
        {
            return RegisterHotKey(hWnd, id, modifier, key);
        }

        public bool Unregiser()
        {
            return UnregisterHotKey(hWnd, id);
        }
    }


    public static class Constants
    {
        //modifiers
        public const int NOMOD = 0x0000;
        public const int ALT = 0x0001;
        public const int CTRL = 0x0002;
        public const int SHIFT = 0x0004;
        public const int WIN = 0x0008;

        //windows message id for hotkey
        public const int WM_HOTKEY_MSG_ID = 0x0312;
    }
}
```

Appendix E

Test case	Pass or fail	Comments
1. Enable recoding by clicking the recode button and spell a word like router. Then stop the recoding by clicking the stop button and verify whether the spelled word was converted to text correctly or not.	Pass	No
2. Enable recoding by clicking the recode button and spell a word like router. Then stop the recoding by clicking the stop button and verify whether the spelled word was recoded in the intermediate wav file without background noise or not.	Pass	This was verified using math lab code attached in the testing chapter and Resuls were attached as graphs in that chapter.
3. Enable recoding by clicking the recode button and spell a word like router. Then stop the recoding by clicking the stop button. After that click do button by selecting search as the action. Verify whether it shows the correct cluster or not.	Pass	No
4. Enable recoding by clicking the recode button and spell a word like router. Then stop the recoding by clicking the stop button. After that click do button by selecting search as the action. Verify whether it shows searched results in the web browser or not.	Pass	For this test case locally hosted wamp web server was used with dummy web sites.
5. Spell two words and check whether the second word is predicted correctly based on the first word.	Pass	Initially meta data loaded to markov model and spelled words from them.