

**A
MAJOR PROJECT REPORT
ON
“Intelligent Virtual Voice Assistant”
submitted in the partial fulfillment of the requirement for the
award of degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE & ENGINEERING**



**Submitted By:
Rishi (A40318052)
Rishabh(A40318089)
Yugant (A40318081)
Shatrughan (A40318047)**

Batch: 2018-2022

**Project Guide:
Dr. Banita (A.P., CSE Deptt.)**

**Department of Computer Science &Engineering
Faculty of Engineering & Technology
P.D.M. University, Bahadurgarh**

DECLARATION

I hereby declare that the work presented in this report entitled “**Intelligent Virtual Voice Assistant**”, in fulfillment of the requirement for the award of the degree Bachelor of Technology in Computer Science & Engineering, submitted in CSE Department, PDMU, Bahadurgarh, Haryana is an authentic record of my own work carried out during my degree under the guidance of **Dr. Banita.**

The work reported in this has not been submitted by me for award of any other degree or diploma.

Date: 24/11/2021

Place: Bahadurgarh, Haryana

Rishi Thakur(A40318052)

Rishabh Kumar(A40318089)

Yugant(A40318081)

Shatrughan(A40318047)

Certificate of Completion

This is to certify that the Project work entitled “**Intelligent Virtual Voice Assistant**” submitted by **Rishi (A40318052), Rishabh (A40318089), Yugant(A40318081), Shatrughan (A40318047)** in fulfillment for the requirements of the award of **Bachelor of Technology** Degree in Computer Science & Engineering at PDMU, Bahadurgarh, Haryana is an authentic work carried out by them under my supervision and guidance. To the best of my knowledge, the matter embodied in the project has not been submitted to any other University / Institute for the award of any Degree.

Dr. Banita
A.P. in CSE Department
Faculty of Engineering & Technology

ACKNOWLEDGEMENT

I express my sincere gratitude to **Dr. Jasvinder Kaur** (Head, Department of CSE) and **Dr. Banita** (A.P., CSE Deptt.), for her valuable guidance and timely suggestions during the entire duration of my project work, without which this work would not have been possible. I would also like to convey my deep regards to all other faculty members who have bestowed their great effort and guidance at appropriate times without which it would have been very difficult on my part to finish this work. Finally, I would also like to thank my friends for their advice and pointing out my mistakes.

ABSTRACT

Use of personal computers (PCs) and smartphones with different types of sensors in them has digitized user's activities in real world. PC applications such as time & date, sending email, keeping notes and searching web gives a view into user's activities. To ease the work of the user and its interaction with the PCs Intelligent Virtual Assistants are being developed these days.

In this thesis we see the approach to design and develop an intelligent voice recognition personal assistant that reduces the utilization of various input devices like keyboard and mouse with PC. Accepting commands via speech makes the software more user friendly and efficient. This thesis contains the representation model of the system, along with the implementation details. It shows details about the addition of new commands and additional features of remote access. This thesis is a technical brief on various Virtual Assistant technology available along with its opportunities and challenges in different areas. The software is named Personal Assistant with Voice Recognition Intelligence as it takes input from the user in form of voice or text, process it and returns the output in various forms like action to be performed or the search result is dictated to the end user. The system is designed in such a way that all the services provided by the computer are accessible by the end user on the user's voice commands.

The goal of the project is to design and implement an intelligent voice recognition personal assistant software that understands the task (given as i/p in audio format), able to divide the task into multiple tasks and perform them accordingly. Along with these various existing applications in this area are also compared based on available features and functionalities supported by them.

LIST OF FIGURES

Figure 2.1. Speech to Text Recognition (STT)

Figure 2.2. Structure of standard Speech Recognition System

Figure 2.3. Hidden Markov Model

Figure 5.1 Weather forecast implementation

Figure 5.2 Sequence Diagram for Use case - Weather Forecast

Figure 5.3 Sequence Diagram for Use case - Weather Forecast

Figure 5.4 Architectural Representation of the system

Figure 5.5 System functioning- Block Diagram

Figure 5.6 Data Flow Diagram of the system

Figure 8.1. Output 1 screenshot

Figure 8.2. Output 2 screenshot

Figure 8.3. Output 3 screenshot

Figure 8.4. Output 4 screenshot

Figure 8.5. Output 5 screenshot

Figure 8.6. Output 6 screenshot

Figure 8.7. Output 7 screenshot

List of Tables

Table- 1: Comparison between voice assistants

Table- 2: Comparison of VPAs like Google now, Cortana and SIRI

Table of Contents

Declaration	2
Acknowledgement...	3
Certificate...	4
Abstract...	5
List of Figures...	6
List of Tables...	7
 CHAPTER 1: Introduction	
1.1 Problem Statement...	11
1.2 Motivation...	12
1.2.1 Information overload on the web...	12
1.2.2 Silos of information...	12
1.2.3 Using multiple applications to get one user task done	13
1.3 Organization of thesis.	13
 CHAPTER 2: Literature Survey	
2.1 Reviewing Existing Solution...	16
2.2 SPEECH-TO-TEXT (STT) or Speech Recognition.	19
2.2.1 Application of Speech-To-Text.....	
2.2.2 Structure of standard Speech Recognition System...	20
2.2.3 Algorithm for Speech-To-Text.....	
2.3 TEXT-TO-SPEECH (TTS).	23
 CHAPTER 3: Software Requirement Specification (SRS)	
3.1 Introduction	26
3.1.1 Purpose	26
3.1.2 Scope	26
3.1.3 Definitions and Abbreviations...	27
3.1.4 References...	27
3.2 Overall Description...	27

3.2.1 Product Perspective	27
3.2.2 Product Functions	27
3.2.3 Operating Environment...	28
3.3 Specific Requirements...	28
3.3.1 External Interface Requirement...	28
3.3.2 Functional Requirements...	29
3.3.3 Non-Functional Requirements...	29
3.4 Conclusion.	30

CHAPTER 4: RESEARCH & DESIGN METHODOLOGY

4.1 Research on Source of Data.	31
4.2 Overview of Technologies Used...	31
4.2.1 Web Applications	31
4.2.2 Scripting Languages...	32
4.2.3 Advantages of javas	32
4.2.4 Disadvantages of javascript...	33
4.3 Python...	34

CHAPTER 5: IMPLEMENTATION OF VIRTUAL ASSISTANT

5.1 Python Libraries Used...	36
5.2 System Architecture	49
5.3 Mathematical Representation...	50
5.4 Functions...	51
5.5 Data Flow Diagram...	51

CHAPTER 6: Constraints and Limitations of the System

6.1 Personal assistant for specific domains...	53
6.2 Sharing task decomposition with other users...	55
6.3 Ability to add new facts and task decomposition to database	56
6.4 Learning from user input and actions...	56
6.5 Personal information on the PCs...	56

CHAPTER 7: Conclusion and Future work	
7.1 Personal Assistants are the future	58
7.2 Advantages of Virtual Assistant...	58
7.3 Supporting tasks...	59
7.4 Building the Agent.	59
7.5 Knowledge Databases...	60
7.6 Planning Algorithms	60
 APPENDIX A : OUTPUTS	 61
APPENDIX B : CODE...	65
References...	69

CHAPTER 1

INTRODUCTION

1.1 Problem Statement

This project aims to serve the following objectives:

1. Main objective of this thesis is to show feasibility of building a personal assistant software (a smart agent) using data sources available on the web, user generated content, data from the sensors of user's PCs and providing knowledge from knowledge databases..
2. To design a smart agent that has contextual information about the user and helps in managing and planning tasks, using web technologies and open data available on the Internet. Contextual information about the user can be location, current time, calendar appointments, relation between tasks, decomposition of tasks, past history of tasks, user choices, interests, likes etc.
3. Agent can use data gathered about the user as well as environment data to better understand what each of the tasks mean and decompose the tasks based on sequence of steps stored in its knowledge base and then plan individual tasks.
4. Building a virtual assistant in order to help regular customers with day to day activities by using commands that can be customized as per users preferences
5. Implementing pre-existing text to speech and speech to text algorithms using the functional modules so developed.

Planning part of the agent will strive to optimize resources and try to improve productivity of the user. It can be used as a time management application as well as a task management application. By combining, related tasks together that can be completed at the same time and around the same location, agent will optimize the user's resources to complete these tasks.

A feedback loop from the user will help the agent to make decisions when there are multiple paths and agent does not have sufficient information to make those decisions.

Assumptions, limitations and constraints in the solution will be highlighted and any additional infrastructure necessary as a complement to the system will be identified.

1.2 Motivation

1.2.1 Information overload on the web

Huge amount of information is being generated by online websites and is primarily consumed by users browsing the website. Websites have used technologies to render the content on the user's screen and let the user make decisions based on his interpretation of that content.

In order to be discovered by the search engines, web sites include some metadata about the content, enabling discovery of the web site's pages when a user makes a relevant query. This model puts user at the center to identify tools such as search engines in order to get to the relevant content for his need, filter from the list of available content and then read the content - making the task of consuming this information increasingly difficult and leading to scalability limitations on the amount of content that can be consumed by users, causing information overload.

Availability of this information coupled with its overload on users has created a need for applications that can act as a conduit to interface users to the digital world by using the online information for the benefit of its users, with a minimal intervention from the users.

1.2.2 Silos of information

Adoption of proprietary structure for the data by enterprises has led to proliferation and creation of silos of information that makes interpretation of this data difficult by other applications interested in that data. Even though large amounts of data and services are available, need for interpreting proprietary formats of the data makes it difficult to easily consume this data and so limits the spread of this information to interested parties who can provide richer value added services.

In contrary, adoption of standard specifications to represent this data and publishing the content and the metadata used to generate the content can enable creation of applications that can interpret this data on behalf of the users and make the consumption of the content easier as well as timely.

1.2.3 Using multiple applications to get one user task done

Proliferation of mobile devices and mobile apps running on these devices has led to specialized applications with a razor focus on helping user accomplish a very specific task, using some of the contextual information available from sensors of mobile device, but with almost no interactions with other applications that user may be using to accomplish other tasks.

This leaves the burden of breaking the tasks for a user's intention to complete a task solely on the user. In a way, user needs to manually manage the workflow of the system as well as data transformation and data flow by using multiple sets of applications to complete one task.

For example, a user trying to make a travel plans need to check for airport codes for nearby airports and then check travel sites for tickets between combinations of airports to reach the destination.

1.3 Organization of the thesis

Thesis report is organized into seven chapters, from introducing the topic to current solutions, Literature review, research and design methodologies, design and implementation of the system to summary of the achievements at the end of the thesis report. Each of the chapter summaries is described below.

Chapter 1: Introduction

This chapter provides introduction to the topic of a smart agent or personal assistant software, motivations and drivers for writing this thesis, objectives or goals set out at the beginning of the thesis. It also covers organization of the thesis, with a brief summary about the contexts of each chapter.

Chapter 2 : Literature Survey

This chapter cover all the theoretical details gathered, knowledge about existing algorithms, books referred and various technologies available before designing and implementing the intelligent virtual voice assistant system.

Chapter 3: System Requirement Specifications (SRS)

Chapter 4: Research & Design Methodologies

This chapter covers research methodology to research the topic as well as software methodology used to build and test the agent. It also covers details about current market trends and solutions in form of applications such as SIRI, Cortana, ReQall, and Google Glass that help user's interactions with the digital world. It covers high-level use cases handled by these software applications.

Chapter 4 of the thesis provides an overview of different technologies that can be leveraged to tap into data sources on the web, as well as some of the tools that could be leveraged to wrap these technologies to build an agent system. It explores the specifications, ontologies, knowledge databases, and online web services that provide access to data or services pertaining to it.

Chapter 5: Design and Implementation of Virtual Assistant

Chapter on designing the agent begins by identifying high-level requirements of the system, dives into more detailed use cases to highlight specific cases of the instances these requirements will be utilized. High-level requirements define the scope of the system pertaining to this thesis and use cases offer a way a path to utilize certain features of the system.

The chapter goes into defining the modules and each module's features, interfaces and interactions with other modules within the system.

This chapter uses sequence diagrams and data flow diagrams to show how the control and information is passed between various modules. Sequence diagrams also highlight decision points within the process such as how the agent uses knowledge database to identify whether the task can be automated or is a manual task. The design identifies logic embedded within the agent that works with the data stored in linked databases and knowledge databases.

Chapter 6: Assumptions, Constraints and Limitations of the system and ways to address them

This chapter reviews the assumptions used while building the system, and also looks at the limitations of the system. It provides improvements to the system to overcome these limitations as well as to relax some of the assumptions and preconditions that are necessary for the current version to work.

Chapter 7: Conclusion and Future work

This chapter discusses how far the thesis assumptions and methodologies used with the help of knowledge databases proved to be useful in building the agent. It discusses catalysts that would enable further spread of the semantic web and related technologies and how this spread can fuel building such agents in future.

CHAPTER 2

LITERATURE SURVEY

2.1 1 Reviewing Existing Solutions

Review of existing solutions such as SIRI, Google Glass, ReQall etc., gave insights into features of these applications as well as deeper understanding of use cases supported by these systems. Purpose of this exercise was to identify the scope – use cases supported by these applications as well as use cases that were beyond the scope due to the technology choices made for the implementation of the system. It also provided references to data sources that could be potentially useful in our implementation of the agent. A better understanding of the technology and design choices were helpful in understanding the impact when these applications are adopted in the real world.

Function	Siri	iFly
Call Service	Yes	Yes
SMS Message Service	Yes	Yes
Open Application	No	Yes
Web Search Service	Google Search Engine	Baidu Search Engine
Reminder	24h	Unlimited
Music Play	Local Library	Local + Remote Library
Command Text Modify	Yes	No
Language	English & French & German & Japanese	Chinese

Table- 1 Comparison

Here is comparison based on features between most popular and commonly used three VPAs which are, Google Now, Siri, Cortana.

	Siri	Cortana	Google Now
Voice	X		
Voice Activation			X
Touch Activation	X	X	X
Interface			X
Gathering information	X	X	
Unique abilities	X	X	
Totals	4	3	3

Table- 2 Comparison

2.2 2 SPEECH-TO-TEXT (STT) or SPEECH RECOGNITION

- Speech recognition is the interdisciplinary sub-field of computational linguistics that develops methodologies and technologies that enables the recognition and translation of spoken language into text by computers. It is also known as "automatic speech recognition" (ASR), "computer speech recognition", or just "speech to text" (STT). It incorporates knowledge and research in the linguistics, computer science, and electrical engineering fields.
- Speech recognition software is a type of software that effectively takes audio content and transcribes it into written words.
- This type of speech-to-text software is extremely valuable to anyone who needs to generate a lot of written content without a lot of manual typing.
- It is also useful for people with disabilities that make it difficult for them to use a keyboard.

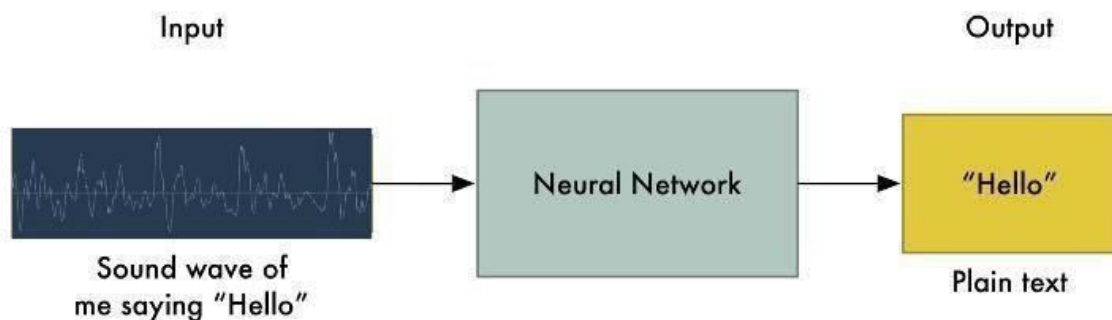


Figure 2.1 Input to Output Wave

2.2.1 Application of Speech-To-Text

- Hands-free computing
- Home automation
- Interactive voice response
- Mobile telephony, including mobile email
- In-car systems, Health care, Military
- Telephony and other domains

2.4.2 Structure of standard Speech Recognition System

The structure of a standard speech recognition system is illustrated in Figure 2. The elements are as follows: Raw speech. Speech is typically sampled at a high frequency, e.g., 16 KHz over a microphone or 8 KHz over a telephone.— This yields a sequence of amplitude values over time. Signal analysis.

Raw speech should be initially transformed and compressed, in order to simplify subsequent processing.— Many signal analysis techniques are available which can extract useful features and compress the data by a factor of ten without losing any important information. Among the most popular:

- Fourier analysis (FFT) yields discrete frequencies over time, which can be interpreted visually. Frequencies are often distributed using a Mel scale, which is linear in the low range but logarithmic in the high range, corresponding to physiological characteristics of the human ear.
- Perceptual Linear Prediction (PLP) is also physiologically motivated, but yields coefficients that cannot be interpreted visually.
- Linear Predictive Coding (LPC) yields coefficients of a linear equation that approximate the recent history of the raw speech values.

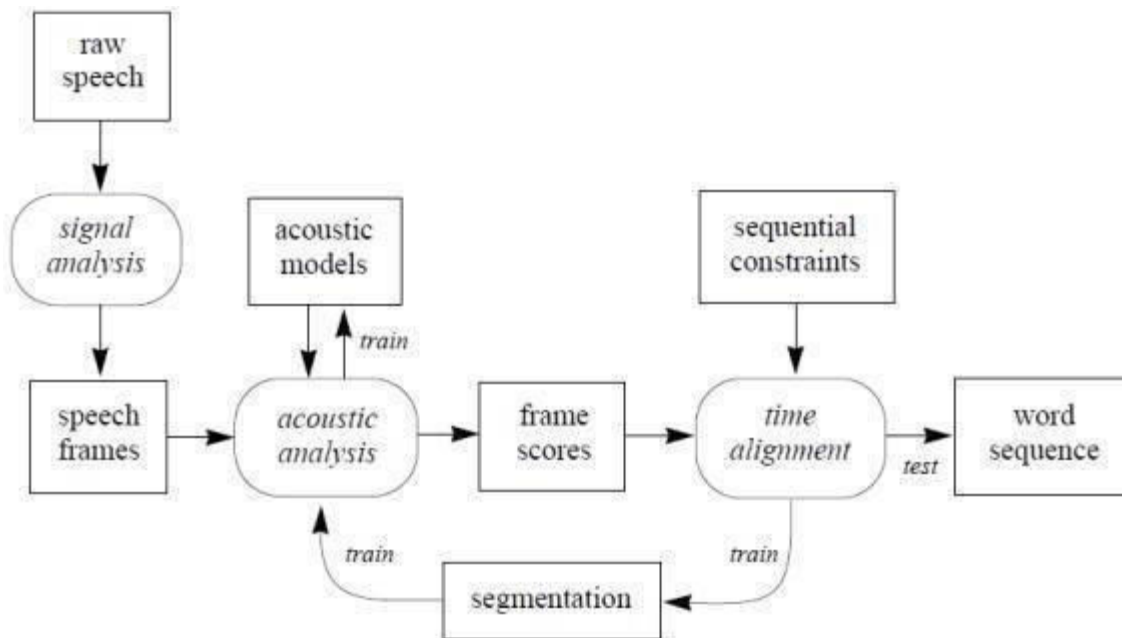


Fig. 1 Structure of standard Speech Recognition System

Figure 2.2 Structure of Speech Recognition

System

2.4.3 Algorithm for Speech-To-Text

There are mainly 2 algorithms that are used for SST. Those are given below:

- Hidden Markov Model(HMM)
- Dynamic Time Warping(DTW)

HIDDEN MARKOV MODEL (HMM) :

A hidden Markov model (HMM) is a statistical Markov model in which the system being modelled is assumed to be a Markov process with unobserved (hidden) states. An HMM can be presented as the simplest dynamic Bayesian network. A Hidden Markov Model is a collection of states connected by transitions, as illustrated in Figure. It begins in a designated initial state. In each discrete time step, a transition is taken into a new state, and then one output symbol is generated in that state. The choice of transition and output symbol are both random, governed by probability distributions. The HMM can be thought of as a black box, where the sequence of output symbols generated over time is observable, but the sequence of states visited over time is hidden from view. This is why it's called a Hidden Markov Model.

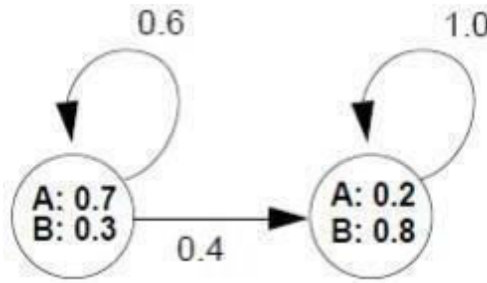


Figure 2.3 Hidden Markov Model

HMMs have a variety of applications. When an HMM is applied to speech recognition, the states are interpreted as acoustic models, indicating what sounds are likely to be heard during their corresponding segments of speech; while the transitions provide temporal constraints, indicating how the states may follow each other in sequence. Because speech always goes forward in time, transitions in a speech application always go forward.

Algorithms of HMM

There are three basic algorithms associated with Hidden Markov Models:

- Forward algorithm, useful for isolated word recognition
- Viterbi algorithm, useful for continuous speech recognition
- Forward-backward algorithm, useful for training an HMM.

Limitations of HMM

- Constant observation of frames
- The Markov assumption
- Lack of formal methods for choosing a model topology
- Large amounts of training data required
- Weak duration modelling
- Restricted output PDFs
- The assumption of conditional independence

DYNAMIC TIME WARPING (DTW) :

The simplest way to recognize an isolated word sample is to compare it against a number of stored word templates and determine which the “best match” is. This goal is complicated by a number of factors. First, different samples of a given word will have somewhat different durations. This problem can be eliminated by simply normalizing the

and the unknown speech so that they all have an equal duration. However, another problem is that the rate of speech may not be constant throughout the word; in other words, the optimal alignment between a template and the speech sample may be nonlinear. Dynamic Time Warping (DTW) is an efficient method for finding this optimal nonlinear alignment. Dynamic time warping (DTW) is a well-known technique to find an optimal alignment between two given (time-dependent) sequences under certain restrictions intuitively; the sequences are warped in a nonlinear fashion to match each other. Originally, DTW has been used to compare different Speech patterns in automatic speech recognition.

In fields such as data mining and information retrieval, DTW has been successfully applied to automatically cope with time deformations and different speeds associated with time- dependent data. In time series analysis, dynamic time warping (DTW) is an algorithm for measuring similarity between two temporal sequences which may vary in time or speed. For instance, similarities in walking patterns could be detected using DTW, even if one person was walking faster than the other can also detected by DTW.

2.3 TEXT-TO-SPEECH (TTS)

In this point of view Speech Synthesis is becoming one of the most important steps towards improving the human interface to the computer. Speech synthesis is the artificial production of speech. This system is called a speech synthesizer, and it can be implemented in software or hardware products.

A text-to-speech (TTS) system converts normal language text into speech; other systems render symbolic linguistic representations like phonetic transcription into speech. Voice is one of the best alternatives for hours of eye strain involved in reading any document. In case of illiterate people Voice is a better interface rather than Graphic User Interface in English.

For that reason research is being done throughout the world for improving the Human Interface to the computer and one of the best options is the ability of a computer to speak to humans.

Text-To-Speech is a process in which input text is first analyzed, then processed and understood, and then the text is converted in digital audio and then “spoken”. It is a small piece of software, which will speak out the text given to it.

2.3.1 Algorithm for Text-To-Speech

There are mainly 2 algorithms that are used for TTS. Those are given below:

- The syllabification algorithm
- Corpus-based speech synthesis

THE SYLLABIFICATION ALGORITHM

The syllabification algorithm breaks a word such that there are minimum number of breaks in the word. This algorithm dynamically looks for polysyllabic units making up the word, and cross checks the database for units availability, and then breaks the word accordingly. If the polysyllabic units are not there, then the algorithm naturally picks up smaller units. For example, For breaking a word .satara. algorithm looks for unit .satara/. in database, if not found it looks for unit combinations such as .sata/, /ra/., /sa/, /tara/. etc.. And at the end it falls back on phone sequence .s/, /a/, /t/, /a/, /r/, /a/.. In the next approach, the syllabification algorithm breaks a word into monosyllables without checking for its availability in the database. In this case syllabification is done based on standard linguistic rules. If a unit is not found in database it can be substituted by a nearest unit or by silence. Although, syllable based synthesis does not require significant prosodic modification, the prosodic modification that needs to be performed in the context of syllable is significantly different from that of conventional diphone based synthesis.

CORPUS-BASED SPEECH SYNTHESIS

Corpus Method is very popular for its high quality and natural speech output.

The basic idea of corpus based speech synthesis or unit selection is that to use the entire speech corpus as the acoustic inventory and to select the run time from this corpus the longest available strings of phonetic segments that match a sequence of target speech sounds in the utterance to be synthesized thereby minimizing the number of concatenations and reducing the need for signal processing.

One main drawback of Corpus method is relative weighting of acoustic distance measures. It needs large speech database with optimal coverage of target domain which is often the whole language. The word or syllable based approaches give better results only in strictly closed application domain.

CHAPTER 3

SOFTWARE REQUIREMENT SPECIFICATION

3.1 INTRODUCTION

3.1.1 Purpose

The purpose of the project is to develop an Android application that provides an intelligent voice assistant with the functionalities as calling services, message transformation, mail exchange, alarm, event handler, location services, music play service, checking weather, searching engine (Google, Wikipedia), camera, Bing translator, Bluetooth headset support, help menu and Windows azure cloud computing.

This project is focusing on the development intelligent virtual assistant with voice control (recognition, generate and analyze corresponding commands, intelligent responses automatically), Google products and relevant APIs (Google map, Google weather, Google search and etc), Wikipedia API and mobile device references ranging from Speech-To- Text, Text-To-Speech technology, Bluetooth headset support and camera; advanced techniques of Cloud computing, Multi-threading, Adobe Photoshop image editing skills.

3.1.2 Scope

The proposed application is an automated application that will act as a virtual assistant to the user. The user will no longer need to operate PCs manually doing all the tasks and subtasks by themselves. Also the application is open sourced that will definitely help it to develop further to improve accuracy and inclusion or more features. Application will be easily affordable and easy to use.

Following is the list of features implemented:

- Access specific file
- Displays date and time
- Open applications
- Weather forecast
- Search information on Google or Wikipedia

3.1.3 Definitions and Abbreviations

- VA : Virtual Assistant
- PCs : Personal Computers
- HMM : Hidden Markov Model
- DTW : Dynamic Time Warping
- DL : Deep Learning
- CNN : Convolutional Neural Network

3.1.4 References

- Claudio Becchetti, Klucio Prina Ricotti. “Speech Recognition: Theory and C++ Implementation “ 2008 edition
- Software Engineering by Sangeeta Sabharwal [33]

3.2 Overall Description

3.2.1 Product Perspective

The web application developed for providing an efficient way to communicate with the PCs for daily users is fully independent product. Our product is not a part of any other system. The user interacts with the application via speech.

3.2.2 Product Functions

The sole purpose of the developed web application is to provide a communication mechanism between user and PCs and an efficient and easy way to use computers and smartphones. Communication between two parties can be achieved in various forms.

3.2.3 Operating Environment

The web application developed works with python as backend language which is backed up by flask as microframework. The User can independently install & use the application on their personal computers, giving commands as voice input. It has friendly interface where user can toggle between different modes through various voice commands.

3.2.4 Constraints

- The developed application can run under any platform i.e Linux, Unix, Windows, Mac etc.
- The audio capturing device needs good listening capabilities so that speech words can be detected clearly.
- User should never use the application in noisy environment.
- User should not be too far or too close with the audio receiving device otherwise speech command can be falsely detected or may not be detected.
- Any web browser must be installed in user's system still Chrome is preferred.

3.3 Specific Requirements

3.3.1 External Interface Requirement

1. User Interface

The user interface must contain:

- Customizable window
- Mechanism to toggle between different modes of operation
- Different window for different mode
- Explanation of how to use each mode

2. Hardware Interfaces

Hardware interfaces exist in computing systems between many of the components such as the various storage devices, other I/O devices etc. these are following for this project.

- Processor : Intel core 2 duo or higher
- Ram : Greater than or equal to 2GB
- Monitor : 15" Color Monitor
- WebCam : 2MP or higher
- Keyboard and Mouse

III. Software Interfaces

The software is developed with all the basic controls and class provided in Python..

- Operating System : Window 7,10, Ubuntu 18.04, Mac etc
- Developing Tool : Sublime Text, Web browser
- Language : Python 4.6, Javascript
- Web Browser : Chrome, Firefox, etc
- Library (Python) : dlib, opencv, flask, pyaudio, pyttsx3, time, webbrowser, speech_recognition
- Framework : Flask (Python)

3.3.2 Functional Requirements

Selection of Mode

- Introduction : User selects mode of operation of application through voice command.
- Input : User speech is stored in database using pyaudio for analysis.
- Processing : System extracts the words, process them to understand the meaning and execute a list of tasks and subtasks.
- Output : The Output data is displayed and the text is converted back to speech. The application is redirected to speech input mode.

3.3.3 Non Functional Requirement

- I) Performance: Application should work without any lag in speech detection and words extractions, etc. The performance shall depend upon hardware and software components of the system.
- II) Reliability: The application should not crash under any circumstances such as speech not detected or words from speech not extracted in noisy environment.
- III) Portability: The applications should be able to run on any machine with minimum hardware requirements. The project is made in python as backend and will work upon all operating systems.
- IV) Maintainability: The application should be maintainable as long as there are no hardware problems. New features can be incorporated in the application.

3.4 Conclusion

The application so developed is user friendly facility provided to the computer user's so that they could connect to their PCs. It has been designed to automate the manual process of communication with PCs using hardware devices such as keyboard, mouse, etc doing each and every task and subtask manually.

CHAPTER 4

RESEARCH & DESIGN METHODOLOGY

This chapter covers sequence of steps, tools, and processes used to define the scope of the smart agent project, research development tools, research process for looking at existing algorithmic solutions as well as designing and implementing the system.

Various open-source frameworks, platforms and tools were used during the research, review, design and implementation phases of this project and have been covered as part of this thesis in later chapters.

4.1 1 Research on Sources of Data

This project was started on the premise that there is sufficient amount of openly available data and information on the web that can be utilized to build an agent that has access to making intelligent decisions for routine user activities.

So, the first step in the process was to uncover these sources of data that are relevant to the use cases of the smart agent.

4.2 Overview of Technologies Used

4.2.1 Web Application

A web application is any application that uses a web browser as a client. The application can be as simple as a message board or a guest signing book on a website, or as complex as a word processor or a spreadsheet. A web application relieves the developer of the responsibility of building a client for a specific type of computer or a specific operating system. Since the client runs in a web browser, the user could be using an IBM compatible or a Mac. They can be running Windows XP or Windows Vista. They can even be using Internet Explorer or Firefox, though some applications require a specific web browser.

Web applications commonly use a combination of server side script (ASP, PHP, etc) and client side script (HTML, Javascript, etc.) to develop the application. The client side script

deals with the presentation of the information while the server side script deals with all the hard stuff like storing and retrieving the information.

The web application tools that we've used span over Cascading style sheets, JavaScript and Hyper Text Markup Language.

4.2.2 Scripting Languages

A scripting or script language is a programming language that supports scripts, programs written for a special run-time environment that automate the execution of tasks that could alternatively be executed one -by -one by a human operator. Scripting languages are often interpreted (rather than compiled).

JavaScript in this view, is particularly glue code, connecting software components, and embedded in a browser.

4.2.3 Advantages of Javascript

- Speed : Being client-side, JavaScript is very fast because any code functions can be run immediately instead of having to contact the server and wait for an answer.
- Simplicity : JavaScript is relatively simple to learn and implement.
- Versatility : JavaScript plays nicely with other languages and can be used in a huge variety of application. Unlike PHP or SSI scripts, JavaScript can be inserted into any web page regardless of the file extension. JavaScript can also be used inside scripts written in other languages such as Perl and PHP.
- Server Load. Being client -side reduces the demand on the website server. Javascript is executed on the client side :
- This means that the code is executed on the user's processor instead of the web server thus saving bandwidth and strain on the web server
- Javascript is a relatively easy language
- The Javascript language is relatively easy to learn and comprises of syntax that is close to English. It uses the DOM model that provides plenty of prewritten functionality to the various objects on pages making it a breeze to develop a script to solve a custom purpose.
- Javascript is relatively fast to the end user.
- As the code is executed on the user's computer, results and processing is

completed almost instantly depending on the task (tasks in javascript on web pages are usually simple so as to prevent being a memory hog) as it does not need to be processed in the site's web server and sent back to the user consuming local as well as server bandwidth.

- Extended functionality to web pages
- Third party add-ons like Greasemonkey enable Javascript developers to write snippets of Javascript which can execute on desired web pages to extend its functionality. If you use a website and require a certain feature to be included, you can write it yourself and use an add-on like Greasemonkey to implement it on the web page.

4.2.4 Disadvantages of Javascript

- Security Issues
- Javascript snippets, once appended onto web pages execute on client servers immediately and therefore can also be used to exploit the user's system. While a certain restriction is set by modern web standards on browsers, malicious code can still be executed complying with the restrictions set.
- Javascript rendering varies
- Different layout engines may render Javascript differently resulting in inconsistency in terms of functionality and interface. While the latest versions of javascript and rendering have been geared towards a universal standard, certain variations still exist. Website Usability Consultants all over the world make a living on these differences, but it enrages thousands of developers on a daily basis.

4.3 Python

- Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including

object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Interpreted – Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Interactive – You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- Object-Oriented – Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

4.3.1 Advantages of python

- Python comes with a huge amount of inbuilt libraries. Many of the libraries are for Artificial Intelligence and Machine Learning. Some of the libraries are Tensorflow (which is high-level neural network library), scikit-learn (for data mining, data analysis and machine learning), pylearn2 (more flexible than scikit-learn), etc. The list keeps going and never ends.
- For other languages, students and researchers need to get to know the language before getting into ML or AI with that language. This is not the case with python. Even a programmer with very basic knowledge can easily handle python.
- The time someone spends on writing and debugging code in python is way less when compared to C, C++ or Java.
- Libraries, their tutorials, handling of interfaces are easily available online.

What can Python do?

- Python can be used on a servers to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

CHAPTER 5

IMPLEMENTATION OF VIRTUAL ASSISTANT

5.1 The python libraries used

Pyaudio

PyAudio provides Python bindings for PortAudio, the cross-platform audio I/O library. With PyAudio, you can easily use Python to play and record audio on a variety of platforms.

PyAudio is inspired by:

- pyPortAudio/fastaudio: Python bindings for PortAudio v18 API.
- tkSnack: cross-platform sound toolkit for Tcl/Tk and Python.

To use PyAudio, first instantiate PyAudio using `pyaudio.PyAudio()` , which sets up the portaudio system.

To record or play audio, open a stream on the desired device with the desired audio parameters using `pyaudio.PyAudio.open()` . This sets up a `pyaudio.Stream` to play or record audio.

Play audio by writing audio data to the stream using `pyaudio.Stream.write()`, or read audio data from the stream using `pyaudio.Stream.read()`.

Note that in “blocking mode”, each `pyaudio.Stream.write()` or `pyaudio.Stream.read()` blocks until all the given/requested frames have been played/recorded. Alternatively, to generate audio data on the fly or immediately process recorded audio data, use the “callback mode” outlined below.

Use `pyaudio.Stream.stop_stream()` to pause playing/recording, and `pyaudio.Stream.close()` to terminate the stream.

Finally, terminate the portaudio session using `pyaudio.PyAudio.terminate()`

Weather forecast implementation

For weather reports, we have used an API called W-underground. By registering as a developer on the w-underground web site we get an API-key, which is used in the request

that the executable sends to the API. The executable gets the weather information in xml format.

[http://api.wunderground.com/api/"+wunderground_key+"/conditions/q/autoip.xml](http://api.wunderground.com/api/)

This xml file detects your location and gives the weather information of your location only. If we want to get the information of specific city or place then we can use following link, [http://api.wunderground.com/api/"+wunderground_key+"/conditions/q/" + city_name + ".xml"](http://api.wunderground.com/api/)

The xml from API contains information like-

Place

Observation time

Type of date like weather of forecast etc.

Temperature

Wind

Pressure

Dew point

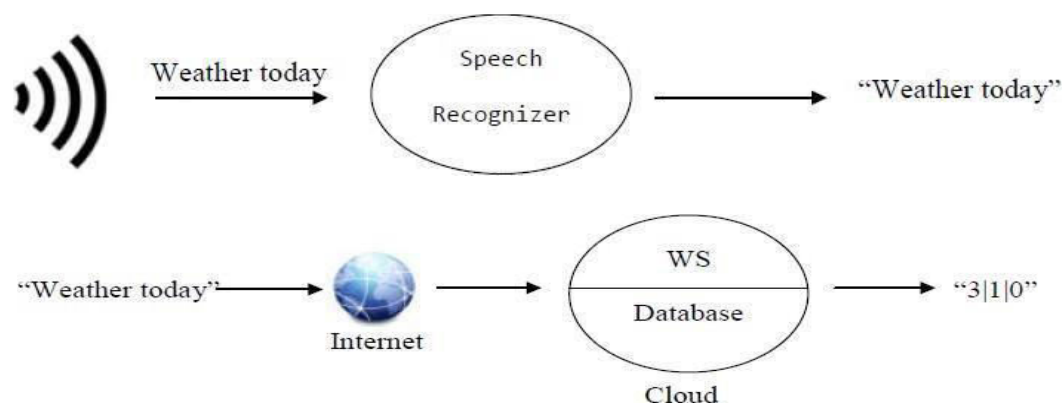


Figure - 5.1 Weather Forecast
Implementation

Sequence Diagram for Use case - Weather Forecast

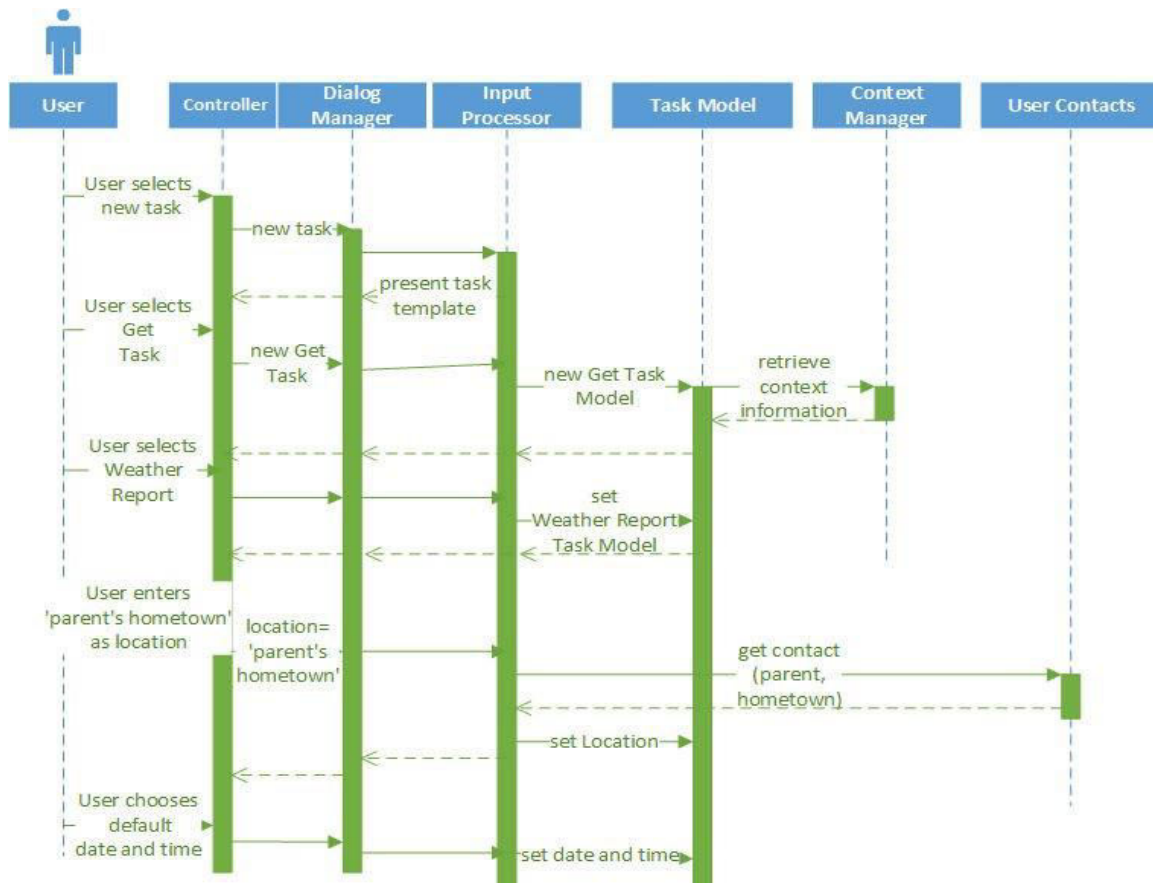


Figure- 5.2 Implementation 1

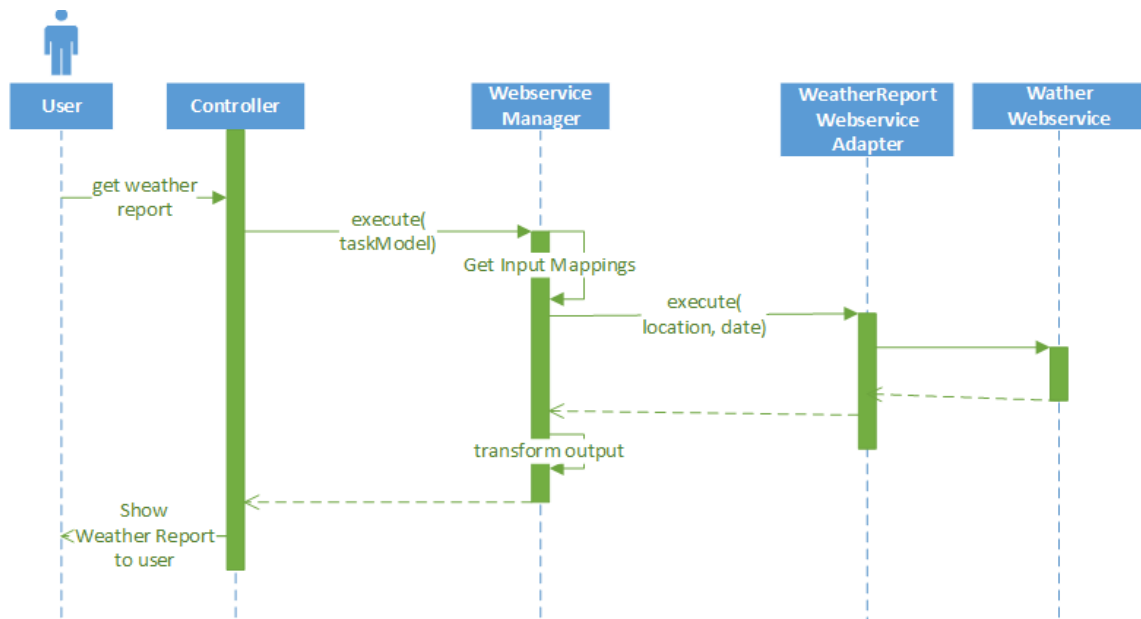


Figure - 5.3 Implementation 2

Web browser

Allows us to open tab in the browser as per our needs, webbrowser – Displays web pages

Purpose: Use the web browser module to display web pages to your users.

Available In: 2.1.3 and later

The web browser module includes functions to open URLs in interactive browser applications. The module includes a registry of available browsers, in case multiple options are available on the system. It can also be controlled with the BROWSER environment variable.

Simple Example

To open a page in the browser, use the open() function.

```
import webbrowser

webbrowser.open('http://docs.python.org/lib/module-webbrowser.html')
```

The URL is opened in a browser window, and that window is raised to the top of the window stack. The documentation says that an existing window will be reused, if possible, but the actual behavior may depend on your browser's settings. Using Firefox on my Mac, a new window was always created.

Windows vs. Tabs

If you always want a new window used, use open_new().

```
import webbrowser

webbrowser.open_new('http://docs.python.org/lib/module-webbrowser.html')
```

If you would rather create a new tab, use open_new_tab() instead.

Using a specific browser

If for some reason your application needs to use a specific browser, you can access the set of registered browser controllers using the get()function. The browser controller has methods to open(), open_new(), and open_new_tab(). This example forces the use of the lynx browser:

```
import webbrowser

b = webbrowser.get('lynx') b.open('http://docs.python.org/lib/module-webbrowser.html')
```

Refer to the module documentation for a list of available browser types.

BROWSER variable

Users can control the module from outside your application by setting the environment variable `BROWSER` to the browser names or commands to try. The value should consist of a series of browser names separated by `os.pathsep`. If the name includes `%s`, the name is interpreted as a literal command and executed directly with the `%s` replaced by the URL. Otherwise, the name is passed to `get()` to obtain a controller object from the registry.

For example, this command opens the web page in lynx, assuming it is available, no matter what other browsers are registered.

```
$ BROWSER=lynx python webbrowser_open.py
```

If none of the names in `BROWSER` work, `webbrowser` falls back to its default behavior.

Command Line Interface

All of the features of the `webbrowser` module are available via the command line as well as from within your Python program.

```
$ python -m webbrowser
```

```
Usage:/Library/Frameworks/Python.framework/Versions/2.5/lib/python2.5/webbrowser.p
|-t] url
```

```
-n: open new window
```

```
-t: open new tab
```

TIME

Takes the time from the system clock and reports it. This module provides various time-related functions. For related functionality, see also the [datetime](#) and [calendar](#) modules.

Although this module is always available, not all functions are available on all platforms. Most of the functions defined in this module call platform C library functions with the same name. It may sometimes be helpful to consult the platform documentation, because the semantics of these functions varies among platforms.

An explanation of some terminology and conventions is in order.

- The epoch is the point where the time starts. On January 1st of that year, at 0 hours, the “time since the epoch” is zero. For Unix, the epoch is 1970. To find out what the epoch is, look at `gmtime(0)`.

- The functions in this module do not handle dates and times before the epoch or far in the future. The cut-off point in the future is determined by the C library; for Unix, it is typically in 2038.
- Year 2000 (Y2K) issues: Python depends on the platform's C library, which generally doesn't have year 2000 issues, since all dates and times are represented internally as seconds since the epoch. Functions accepting a `struct_time` (see below) generally require a 4-digit year. For backward compatibility, 2-digit years are supported if the module variable `accept2dyear` is a non-zero integer; this variable is initialized to 1 unless the environment variable `PYTHONY2K` is set to a non-empty string, in which case it is initialized to 0. Thus, you can set `PYTHONY2K` to a non-empty string in the environment to require 4-digit years for all year input. When 2-digit years are accepted, they are converted according to the POSIX or X/Open standard: values 69-99 are mapped to 1969-1999, and values 0-68 are mapped to 2000-2068. Values 100-1899 are always illegal. Note that this is new as of Python 1.5.2(a2); earlier versions, up to Python 1.5.1 and 1.5.2a1, would add 1900 to year values below 1900.
- UTC is Coordinated Universal Time (formerly known as Greenwich Mean Time, or GMT). The acronym UTC is not a mistake but a compromise between English and French.
- DST is Daylight Saving Time, an adjustment of the timezone by (usually) one hour during part of the year. DST rules are magic (determined by local law) and can change from year to year. The C library has a table containing the local rules (often it is read from a system file for flexibility) and is the only source of True Wisdom in this respect.
- The precision of the various real-time functions may be less than suggested by the units in which their value or argument is expressed. E.g. on most Unix systems, the clock "ticks" only 50 or 100 times a second.
- On the other hand, the precision of `time()` and `sleep()` is better than their Unix equivalents: times are expressed as floating point numbers, `time()` returns the most accurate time available (using Unix `gettimeofday` where available), and `sleep()` will accept a time with a nonzero fraction (Unix `select` is used to implement this, where available).

- The time value as returned by `gmtime()`, `localtime()`, and `strptime()`, and accepted by `asctime()`, `mktime()` and `strftime()`, is a sequence of 9 integers. The return values of `gmtime()`, `localtime()`, and `strptime()` also offer attribute names for individual fields.

Boolean value indicating whether two-digit year values will be accepted. This is true by default, but will be set to false if the environment variable `PYTHON2K` has been set to a non-empty string. It may also be modified at run time.

`time.altzone`

The offset of the local DST timezone, in seconds west of UTC, if one is defined. This is negative if the local DST timezone is east of UTC (as in Western Europe, including the UK). Only use this if `daylight` is nonzero.

`time.asctime([t])`

Convert a tuple or `struct_time` representing a time as returned by `gmtime()` or `localtime()` to a 24- character string of the following form: 'Sun Jun 2023:21:05 1993'. If `t` is not provided, the current time as returned by `localtime()` is used. Locale information is not used by `asctime()`.

Note: Unlike the C function of the same name, there is no trailing newline.

`time.clock()`

On Unix, return the current processor time as a floating point number expressed in seconds. The precision, and in fact the very definition of the meaning of “processor time”, depends on that of the C function of the same name, but in any case, this is the function to use for benchmarking Python or timing algorithms.

On Windows, this function returns wall-clock seconds elapsed since the first call to this function, as a floating point number, based on the Win32 function `QueryPerformanceCounter`. The resolution is typically better than one microsecond.

`time.ctime([secs])`

Convert a time expressed in seconds since the epoch to a string representing local time. If `secs` is not provided or `None`, the current time as returned by `time()` is used. `ctime(secs)` is equivalent to `asctime(localtime(secs))`. Locale information is not used by `ctime()`.

`time.daylight` Nonzero if a DST timezone is defined.

time.gmtime([secs])

Convert a time expressed in seconds since the epoch to a struct_time in UTC in which the dst flag is always zero. If secs is not provided or None, the current time as returned by time() is used. Fractions of a second are ignored. See above for a description of the struct_time object. See calendar.timegm() for the inverse of this function.

time.localtime([secs])

Like gmtime() but converts to local time. If secs is not provided or None, the current time as returned by time() is used. The dst flag is set to 1 when DST applies to the given time.

time.mktime(t)

This is the inverse function of localtime(). Its argument is the struct_time or full 9-tuple (since the dst flag is needed; use -1 as the dst flag if it is unknown) which expresses the time in local time, not UTC. It returns a floating point number, for compatibility with time(). If the input value cannot be represented as a valid time, either OverflowError or ValueError will be raised (which depends on whether the invalid value is caught by Python or the underlying C libraries). The earliest date for which it can generate a time is platform- dependent.

time.sleep(secs)

Suspend execution for the given number of seconds. The argument may be a floating point number to indicate a more precise sleep time. The actual suspension time may be less than that requested because any caught signal will terminate the sleep() following execution of that signal's catching routine. Also, the suspension time may be longer than requested by an arbitrary amount because of the scheduling of other activity in the system.

time.strftime(format[, t])

Convert a tuple or struct_time representing a time as returned by gmtime() or localtime() to a string as specified by the format argument. If t is not provided, the current time as

returned by `localtime()` is used. `format` must be a string. `ValueError` is raised if any field in `t` is outside of the allowed range.

0 is a legal argument for any position in the time tuple; if it is normally illegal the value is forced to a correct one.

The following directives can be embedded in the format string. They are shown without the optional field width and precision specification, and are replaced by the indicated characters in the `strftime()` result:

Directive	Meaning
<code>%a</code>	Locale's abbreviated weekday name.
<code>%A</code>	Locale's full weekday name.
<code>%b</code>	Locale's abbreviated month name.
<code>%B</code>	Locale's full month name.
<code>%c</code>	Locale's appropriate date and time representation.
<code>%d</code>	Day of the month as a decimal number [01,31].
<code>%H</code>	Hour (24-hour clock) as a decimal number [00,23].
<code>%I</code>	Hour (12-hour clock) as a decimal number [01,12].
<code>%j</code>	Day of the year as a decimal number [001,366].
<code>%m</code>	Month as a decimal number [01,12].
<code>%M</code>	Minute as a decimal number [00,59].
<code>%p</code>	Locale's equivalent of either AM or PM. (1)
<code>%S</code>	Second as a decimal number [00,61]. (2)
<code>%U</code>	Week number of the year (Sunday as the first day of the week)
<code>%w</code>	Weekday as a decimal number [0(Sunday),6].
<code>%W</code>	Week number of the year (Monday as the first day of the week)
<code>%x</code>	Locale's appropriate date representation.
<code>%X</code>	Locale's appropriate time representation.
<code>%y</code>	Year without century as a decimal number [00,99].
<code>%Y</code>	Year with century as a decimal number.
<code>%Z</code>	Time zone name (no characters if no time zone exists).
<code>%% A</code>	literal '%' character.

Here is an example, a format for dates compatible with that specified in the RFC 2822 Internet email standard.

```
>>> from          time          import          gmtime,          strftime
>>> strftime("%a,    %d    %b %Y    %H:%M:%S +0000", gmtime())
'Thu,          28          Jun          2001          14:17:15 +0000'
```

Using pytsx

An application invokes the `pytsx.init()` factory function to get a reference to a `pytsx.Engine` instance. During construction, the engine initializes a `pytsx.driver.DriverProxy` object responsible for loading a speech engine driver implementation from the `pytsx.drivers` module. After construction, an application uses the engine object to register and unregister event callbacks; produce and stop speech; get and set speech engine properties; and start and stop event loops.

The Engine factory

```
pytsx.init([driverName : string, debug : bool]) → pytsx.Engine
```

Gets a reference to an engine instance that will use the given driver. If the requested driver is already in use by another engine instance, that engine is returned. Otherwise, a new engine is created.

Parameters:

- `driverName` –
- Name of the `pytsx.drivers` module to load and use. Defaults to the best available driver for the platform, currently:
- `debug` – Enable debug output or not.

The Engine interface

- `ImportError` – When the requested driver is not found
- `RuntimeError` – When the driver fails to

initialize Raises:

```
classpytsx.engine.Engine
```

Provides application access to text-to-speech synthesis.

`connect(topic : string, cb : callable) → dict`

Registers a callback for notifications on the given topic.

Parameters:

- `topic` - Name of the event to subscribe to.
- `cb` - Function to invoke when the event

fires. Returns:

A token that the caller can use to unsubscribe the callback later. The following are the valid topics and their callback signatures.

started-utterance

Fired when the engine begins speaking an utterance. The associated callback must have the following signature.

`onStartUtterance(name : string) → None`

Parameters: `name` – Name associated with the utterance.

started-word

Fired when the engine begins speaking a word. The associated callback must have the following signature.

`onStartWord(name : string, location : integer, length : integer)`

Parameters: `name` – Name associated with the utterance.

finished-utterance

Fired when the engine finishes speaking an utterance. The associated callback must have the following signature.

`onFinishUtterance(name : string, completed : bool) → None`

Parameters:

- `name` – Name associated with the utterance.
- `completed` – True if the utterance was output in its entirety or not.

error

Fired when the engine encounters an error. The associated callback must have the following signature.

onError(name : string, exception : Exception) → None

Parameters:

- name – Name associated with the utterance that caused the error.
- exception – Exception that was raised.

disconnect(token : dict)

Unregisters a notification callback.

Parameters: token – Token returned by connect() associated with the callback to be disconnected.

endLoop() → None

Ends a running event loop. If startLoop() was called with useDriverLoop set to True, this method stops processing of engine commands and immediately exits the event loop. If it was called with False, this method stops processing of engine commands, but it is up to the caller to end the external event loop it started.

Raises RuntimeError:

When the loop is not running

getProperty(name : string) → object

Gets the current value of an engine property.

Parameters: name – Name of the property to query.

Returns: Value of the property at the time of this invocation.

The following property names are valid for all drivers.

rate

Integer speech rate in words per minute. Defaults to 200 word per minute.

voice

String identifier of the active voice.

voices

List of pytsx.voice.Voice descriptor objects.

volume

Floating point volume in the range of 0.0 to 1.0 inclusive. Defaults to 1.0.

isBusy() → **bool**

Gets if the engine is currently busy speaking an utterance or not.

Returns: True if speaking, false if not.

runAndWait() → **None**

Blocks while processing all currently queued commands. Invokes callbacks for engine notifications appropriately. Returns when all commands queued before this call are emptied from the queue.

say(text : unicode, name : string) → **None**

Queues a command to speak an utterance. The speech is output according to the properties set before this command in the queue.

Parameters:

- text – Text to speak.
- name – Name to associate with the utterance. Included in notifications about this utterance.

5.2 2 System Architecture

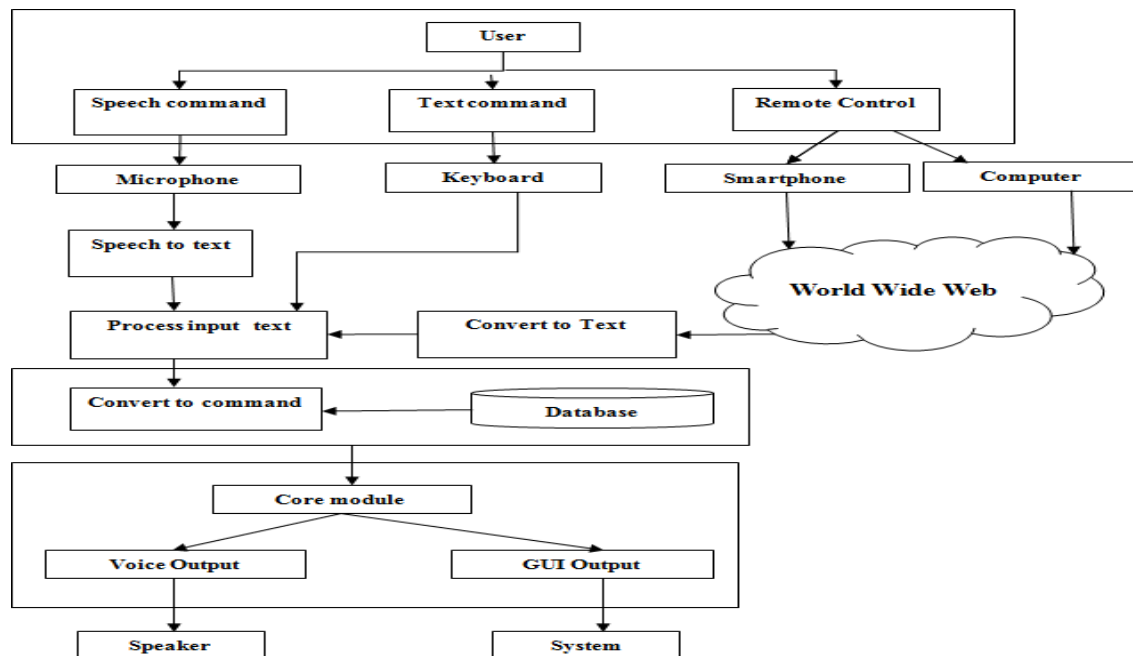


Figure- 5.4 Architecture

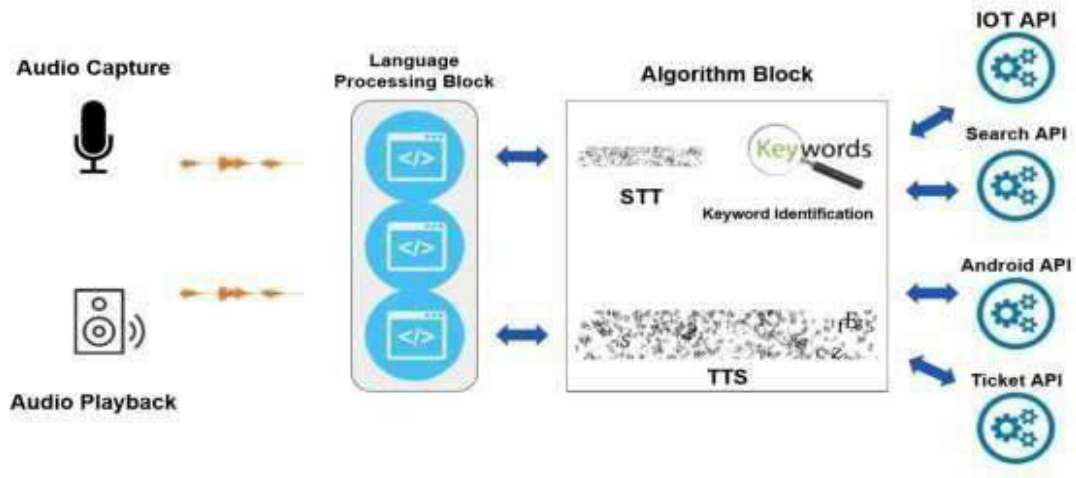


Figure - 5.5 Audio Processing

5.3 MATHEMATICAL REPRESENTATION

Input Set

The personal assistant takes voice input, text input and text input through remote signal.

So the set of inputs will be,

$I_1 = \{\text{predefined command, fixed pattern sentential command, random sentences as command}\}$

$I_2 = \{\text{voice, text, remote text}\}$

Thus, $I = I_1 \cup I_2$ (1)

Input $I = \{\text{all sentences in English via speech, all sentences in English as text, remote text input}\}$

Output Set

The outputs for the desired inputs are response determined by the system according to the input given and the database containing all the necessary inputs and their respective outputs.

$O_1 = \{\text{voice, display, text}\}$

$O_2 = \{\text{GUI, application response}\}$

Thus, $O = O_1 \cup O_2$ (2)

Output $O = \{\text{Response for corresponding voice input, Response for corresponding input via GUI, application response}\}$

There is a one to one relation between input and output. For single input there is only one output.

5.4 Functions

Following are the operations performed on the input in the system:

Recognize()

This operation basically gets the input from the user. For text, the input is saved directly into the database while in case of remote signal, the signal is directly converted to the command. For speech input, it converts the voice into text and saves it.

Extract()

This operation analyses the input string saved and system gets an idea about which command is expected to be executed for the respective input.

Search()

This operation searches the local database for the response of the command extracted by the previous operation.

Response()

This operation gives the output that we see on the screen or through speech for the given command.

5.5 5 Data Flow Diagram

Data Flow Sequence

- a. Initialize device: Initialize the device by calling its name.
- b. Task Manager: Conversion of Speech-to-Text and Text-to-Speech is performed by task manager.
- c. Service Manager: Analysis of commands and matching them with web service adapter and cloud server.
- d. Execute Command: After finding the match for the given command, run the respective python script.

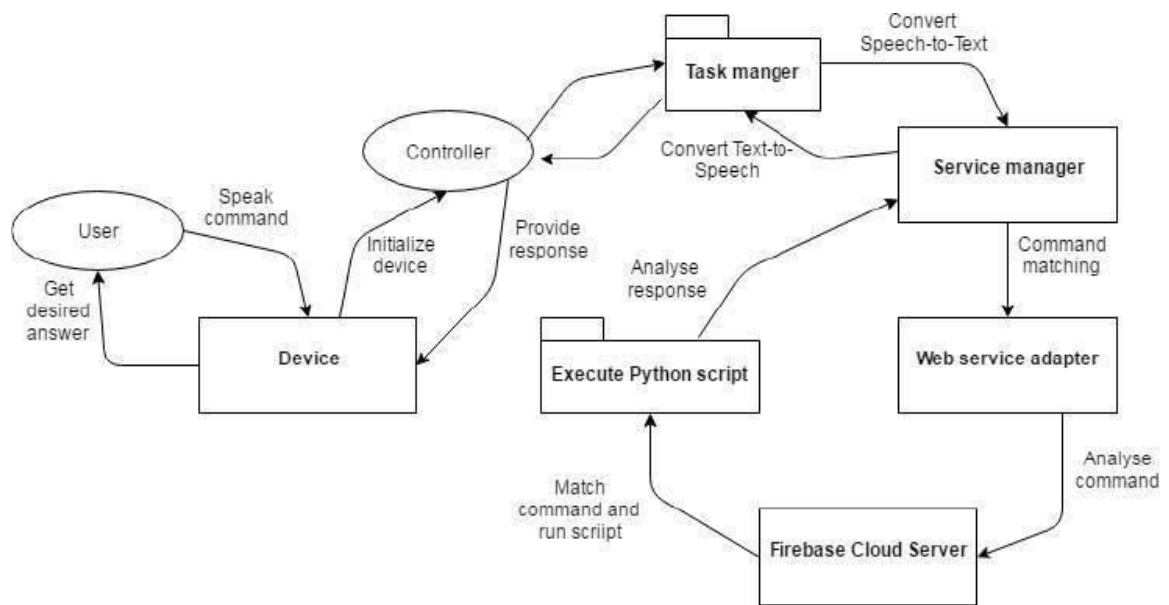


Figure - 5.6 Data Flow Diagram

CHAPTER 6

Constraints and Limitations of the system

This chapter highlights limitations of the architecture and design of the smart agent, and how these limitations impose constraints in terms features provided to the user.

6.1 Personal assistant can be built for specific domains but integrating this with the knowledge of the domain is the challenge.

Knowledge database such as Cyc carries knowledge across multiple domains while the agent is built for particular task domains such as travel, shopping, entertainment etc. This requires partitions within the knowledge base in order to focus on relevant aspects of the knowledge that should be used and interpreted by the agent. Even then, depth of knowledge may be different for different concepts and coupled with level of information needed for the agent, it becomes necessary to review the assertions and relationships contained in the knowledge base when implementing support for a new activity in the agent.

6.2 Sharing task decomposition with other users

Knowledge database provide flexibility in the design of the system, and let the agent be customized based on the user's planning approach to specific tasks. Sharing this knowledge across users can let the best practices be spread to different users, and thus helping formation of community of users sharing their individual knowledge about task planning with others, replicating productive behavior.

6.3 Ability to add new facts and task decomposition to database

In the current design, task domain models use knowledge bases to drive the task planning. But, there is no explicit interface to modify the knowledge base. One of the key features will be to enable the user to add content to knowledge database that can in turn influence how the activities are planned. This will in turn make the agent learn from the user, and become a personalized assistant thru the use of knowledge base.

6.4 Learning from user input and actions

In addition to getting user input for updating the knowledge base, user selections and patterns in tasks can be used by the agent to learn user preferences, affinities based on location and time. This information can then be used to improve the user experience by reducing input selections based on the prior selections of this specific user in similar use contexts.

6.5 Personal information on the PCs

User information is spread across different applications such as address book, emails, call records, calendar etc., on the device. This information needs to be consolidated and provided in a single ontology that can enable applications to access this information in more meaningful ways, with a better understanding of the data they hold.

CHAPTER 7

Conclusion and Future work

7.1 Personal Assistants are the future

Personal assistant software improves user productivity by managing routine tasks of the user and by providing information from online sources to the user. As discussed earlier, technologies such as web services, sharing of data, linked data, shared ontologies, knowledge databases, and mobile devices are proving to be enablers for tools such as personal assistant software.

Building an agent that can replace a human assistant has been a holy grail for software industry, especially in the field of artificial intelligence. Difficulties associated with capturing human intelligence in models that can be used to drive the agent have been one of the primary bottlenecks in building such agents. With the availability of data in semantic form, where the data carries itself the meaning and data sources are interlinked with each other, provides an opportunity to first capture human knowledge in this form and then apply reasoning engines that can interpret these models to make inferences for simple tasks.

This thesis work included conducting research on web technologies, data sources that are available in semantic web form as well as web services, knowledge databases with a view to model simple day-to-day tasks of users using these technologies and to design personal assistant software that can leverage these technologies.

7.2 2 Advantages of Virtual Assistant

- **EASE OF USE:** Virtual assistant created has a very easy to use interface that makes daily tasks much easier and effortless to do.
- **EASY MODIFICATION:** We can add newer functions and capabilities easier as per the needs of the consumer with just the usage of a few elif functions.
- **CUSTOMIZABILITY:** The commands that drive the system can be personalised as per the preferences of the user and can be modified as per the pitch of the users voice.

- **PRIVACY:** Module can be modified so that only few people can have access to this system .

7.3 Possible Future Advancements

- **Recommendations:** Virtual Assistant would provide recommendations based on the integration of anonymised crowd data (e.g. Trip_advisor) with personal data (e.g. personal choices, preferences, interests, debit/credit card use).
- **Push notification:** Virtual Assistant can push notifications to user or intended personnel based on real time events like location, time etc.

7.4 Building the agent

As part of future work, agent will be built based on the design elements in this thesis in order to prove the validity of the design and as part of this process improvements to this design and architecture will be made. A simple and intuitive user interface needs to be built and tested to make the interactions with the agent simple and productive.

7.5 Knowledge Databases

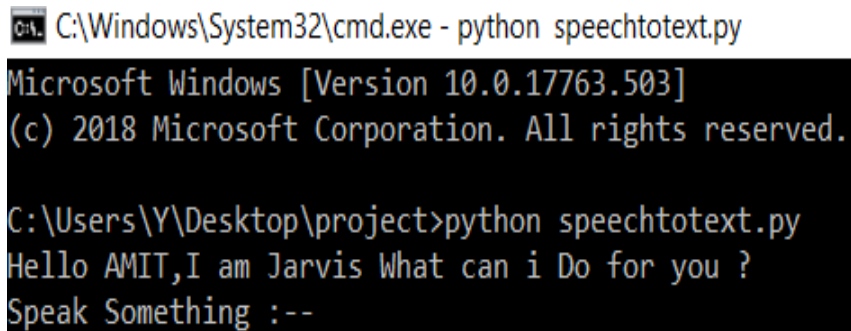
Knowledge databases such as OpenCyc do not provide a reliable SPARQL endpoint for consuming its data in RDF form. Sub-Tasks captured in Cyc are used in the design of the agent. This information for most part is not currently part of the knowledge base and needs to be added into the knowledge base. This information needs to be reviewed for any new task that is supported by the agent.

7.6 Planning Algorithms

Agent should be able to model complex task dependencies and use these models to recommend optimized plans for the user. It needs to be tested for finding optimum paths when a task has multiple sub-tasks and each sub-task can have its own sub-tasks. In such a case there can be multiple solutions to paths, and the agent should be able to consider user preferences, other active tasks, priorities in order to recommend a particular plan.

Agent can prove to be a useful companion for the user if it can take these considerations into account while suggesting plans to the user.

APPENDIX A: OUTPUTS

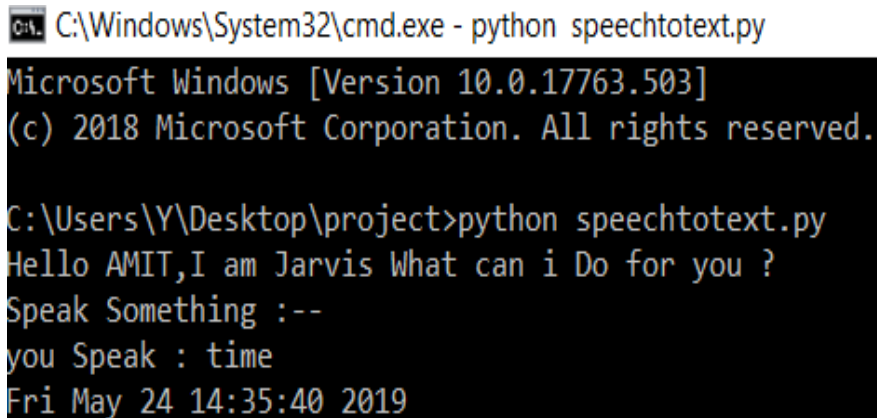


C:\Windows\System32\cmd.exe - python speechtotext.py

Microsoft Windows [Version 10.0.17763.503]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Y\Desktop\project>python speechtotext.py
Hello AMIT,I am Jarvis What can i Do for you ?
Speak Something :--

Figure 8.1



C:\Windows\System32\cmd.exe - python speechtotext.py

Microsoft Windows [Version 10.0.17763.503]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Y\Desktop\project>python speechtotext.py
Hello AMIT,I am Jarvis What can i Do for you ?
Speak Something :--
you Speak : time
Fri May 24 14:35:40 2019

Figure 8.2

```
C:\Windows\System32\cmd.exe - python speechtotext.py
Microsoft Windows [Version 10.0.17763.503]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Y\Desktop\project>python speechtotext.py
Hello AMIT,I am Jarvis What can i Do for you ?
Speak Something :--
you Speak : search
Enter the search term: nsit delhi
```

Figure 8.3

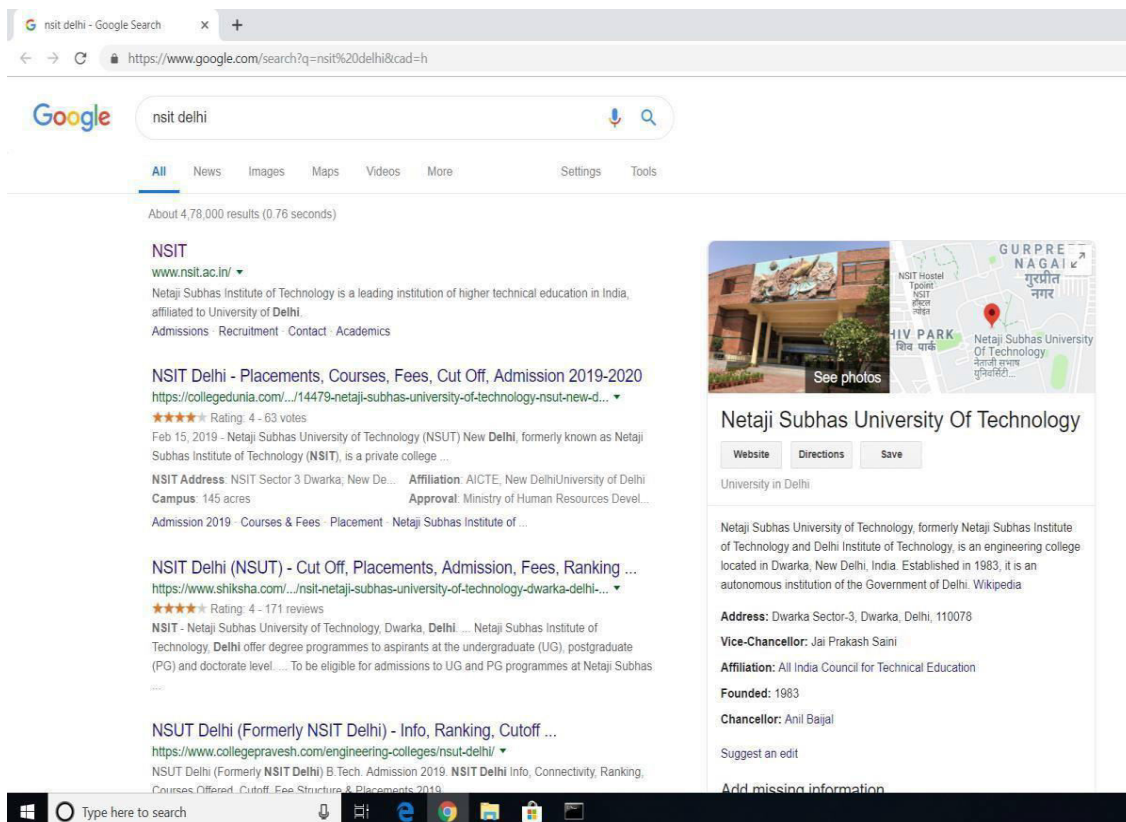


Figure 8.4

```
C:\Windows\System32\cmd.exe - python spechtotext.py
Microsoft Windows [Version 10.0.17763.503]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Y\Desktop\project>python spechtotext.py
Hello AMIT,I am Jarvis What can i Do for you ?
Speak Something :--
you Speak : weather
Location :dwarka
clouds 40
temperature 304.15
country IN
humidity 45
pressure 1006
windspeed 5.1
sunset 1558705222
sunrise 1558655831
```

Figure 8.5

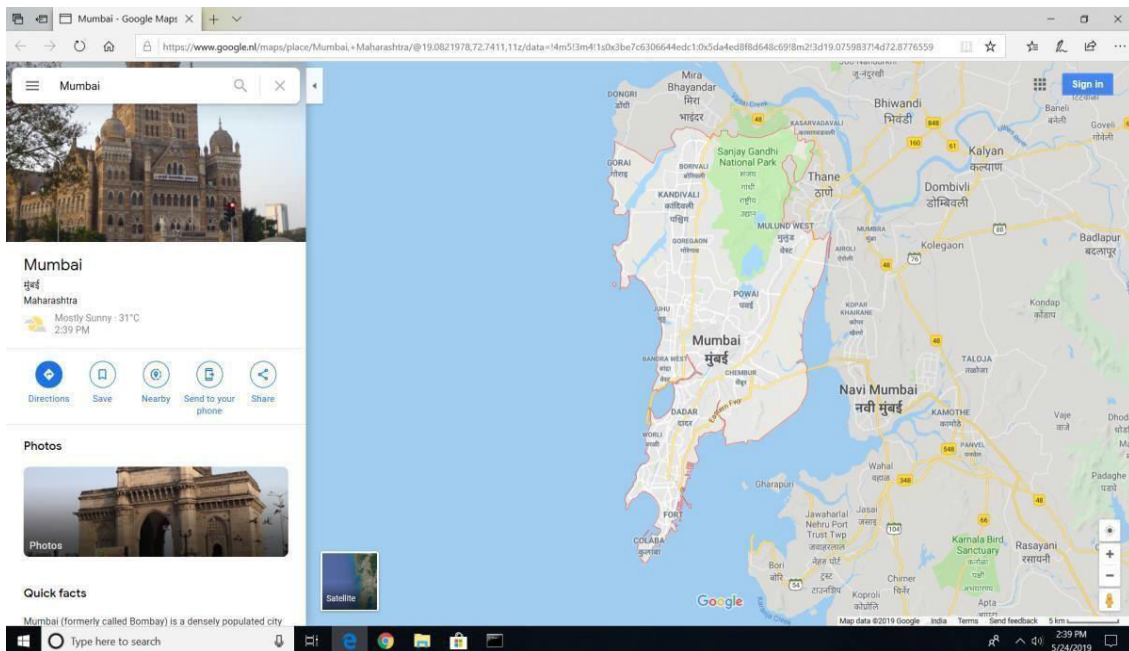
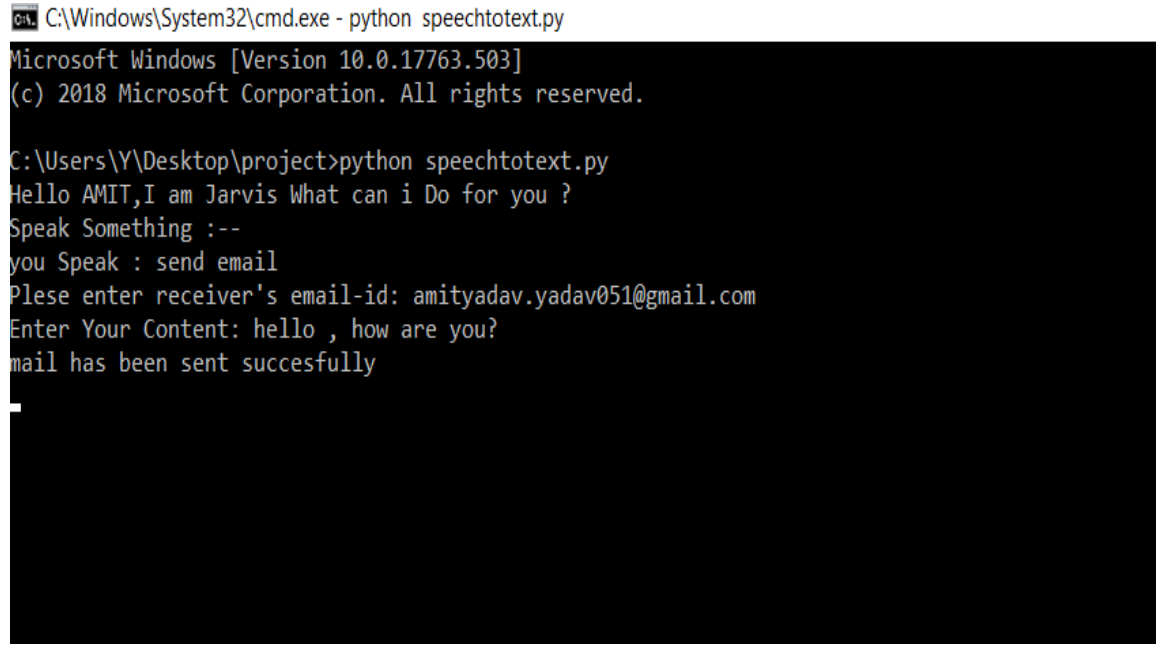


Figure 8.6



```
C:\Windows\System32\cmd.exe - python speechtotext.py
Microsoft Windows [Version 10.0.17763.503]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Y\Desktop\project>python speechtotext.py
Hello AMIT,I am Jarvis What can i Do for you ?
Speak Something :--
you Speak : send email
Plese enter receiver's email-id: amityadav.yadav051@gmail.com
Enter Your Content: hello , how are you?
mail has been sent succesfully
```

Figure 8.7

APPENDIX B: CODE

#Python Libraries Imported

```
import pyaudio
#import pyttsx 3
import time
import threading
import webbrowser as wb
import speech_recognition as sr
r=sr.Recognizer()
import os
from gtts import gTTS
import requests
import re
import webbrowser as wb
from playsound import playsound
```

#Adding text file (city names)

```
def textfile():
    with open(r"C:\Users\Y\Desktop\project\city.txt") as f:
        for line in f:
            if inn in line:
                pb=re.findall("\d+', line)
            return pb;

#eng=pyttsx3.init()
location="INDIA"
counter = 0
chrome_path = "C:/Program Files (x86)/Google/Chrome/Application/chrome.exe %s"
```

Initial input

```
#eng.setProperty('rate',150);
#eng.say("Hello AMIT ,i am Jarvis What can i Do for you ? ");
#eng.runAndWait()
print("Hello AMIT,I am Jarvis What can i Do for you ?");
tts= gTTS(text="Hello AMIT,I am Jarvis What can i Do for you ?", lang='en')
```

```

playsound("pcvoice"+ str(counter) + ".mp3",True)
counter = counter + 1
#os.system("start pcvoice.mp3")
print("Speak Something :--");
#r.energy_threshold = 4000

```

#Different cases for different functions

```

while 1:
    try:
        with sr.Microphone() as source:
            r.adjust_for_ambient_noise(source)
            audio = r.listen(source)
            pinger=r.recognize_google(audio)

    try:
        if(len(pinger)>1):
            print("you Speak : "+pinger);
            #eng.say("you speak "+pinger);
            #eng.runAndWait()

```

#For System Date & Time

```

        if ("time" in pinger):
            tts= gTTS(text=time.ctime() , lang='en')
            tts.save("pcvoice"+ str(counter) + ".mp3")
            playsound("pcvoice"+ str(counter) + ".mp3",True)
            counter = counter + 1
            #os.system("start pcvoice.mp3")
            print(time.ctime());
            Continue

```

For Location Services

```

        elif "where is " in pinger:
            pinger=pinger.split(" ")
            if len(pinger)>3:
                location=pinger[2]+" " +pinger[3]
            else:
                location=pinger[2]
            wb.open_new_tab("https://www.google.nl/maps/place/" + location + "/"&";");
            tts= gTTS(text="Location has been successfully displayed on your screen", lang='en')

```

```

tts.save("pcvoice"+ str(counter) + ".mp3")
    playsound("pcvoice"+ str(counter) + ".mp3",True)
    counter = counter + 1
#os.system("start pcvoice.mp3")
#eng.say("Here is "+location);
#eng.runAndWait()
continue

```

Sending an Email

```

elif "send email" in pinger :
    tts= gTTS(text="Please enter receiver's email i d", lang='en')
tts.save("pcvoice"+ str(counter) + ".mp3")
    playsound("pcvoice"+ str(counter) + ".mp3",True)
    counter = counter + 1
#os.system("start pcvoice.mp3")
    mail=raw_input("Please enter receiver's email-id: ");

    tts= gTTS(text="Enter your content" , lang='en')
tts.save("pcvoice"+ str(counter) + ".mp3")
    playsound("pcvoice"+ str(counter) + ".mp3",True)
    counter = counter + 1

cont=raw_input("Enter Your Content: ");
import yagmail
yag = yagmail.SMTP('amityadav.yadav1998@gmail.com','amittatla1998')
yag.send(mail, "Mail BOT" ,contents=cont);
print('mail has been sent successfully')
tts= gTTS(text="Mail has been successfully sent" , lang='en')
tts.save("pcvoice"+ str(counter) + ".mp3")
    playsound("pcvoice"+ str(counter) + ".mp3",True)
    counter = counter + 1
#os.system("start pcvoice.mp3")

```

#Weather Functionality

```

elif "weather" in pinger:
    tts= gTTS(text="Enter the location" , lang='en')
tts.save("pcvoice"+ str(counter) + ".mp3")
    playsound("pcvoice"+ str(counter) + ".mp3",True)
    counter = counter + 1

```

```

inn = raw_input("Location :")
p=requests.get('http://api.openweathermap.org/data/2.5/weather?q='+inn+'&appid
=c6085ac942cb44ce15cf5e8a40707470');
json_obj=p.json();
weather = {}
weather["country"] =json_obj['sys']['country']
weather["clouds"] =json_obj['clouds']['all']
weather["windspeed"] =json_obj['wind']['speed']
weather["temperature"] =json_obj['main']['temp']
weather["pressure"] =json_obj['main']['pressure']
weather["humidity"] =json_obj['main']['humidity']
weather["sunrise"] =json_obj['sys']['sunrise']
weather["sunset"] =json_obj['sys']['sunset']
for k,v in weather.iteritems():
    print k, v
#url="precipitation" ;#"temperature", "wind speed";

```

#For Google Web Search

```

#wb.open_new_tab("https://openweathermap.org/weathermap?basemap=map&cities=true&layer="+
"precipitation+"&lat=28&lon=78&zoom=4");

```

```

elif "search" in pinger:
    tts= gTTS(text="Enter the search term" , lang='en')
    tts.save("pcvoice"+ str(counter) + ".mp3")
    playsound("pcvoice"+ str(counter) + ".mp3",True)
    counter = counter + 1
    search = raw_input("Enter the search term: ")
wb.get(chrome_path).open("http://www.google.com/?#q=" + search);

```

```

else:
    continue
except (LookupError,sr.RequestError) as e :
    continue
except sr.UnknownValueError:
    Continue

```

REFERENCES

10.1 List of References

[1] D. Yu and L. Deng "Automatic Speech Recognition: A Deep Learning Approach" (Publisher: Springer) published near the end of 2014

[2] Claudio Becchetti, Klucio Prina Ricotti. "Speech Recognition: Theory and C++ Implementation" 2008 edition

[3] Juang, B. H.; Rabiner, Lawrence R. "Automatic speech recognition—a brief history of the technology development" (PDF). p. 6. Retrieved 17 January 2015.

[4] Deng, L.; Li, Xiao (2013). "Machine Learning Paradigms for Speech Recognition: An Overview". IEEE Transactions on Audio, Speech, and Language Processing.

[5] P. Milhorat, S. Schlögl, G. Chollet, J. Boudy, A. Esposito and G. Pelosi, "Building The Next Generation Of Personal Digital Assistants", 1st International Conference on Advanced Technologies for Signal and Image Processing - ATSIP 2014 March 17-19, 2014, Sousse, Tunisia

[6] Hong-wen Sie¹, Dau-Cheng Lyu¹, Zhong-Ing Liou¹, Ren-Yuan Lyu^{1, 2}, Yuang-Chin Chiang³¹ Dept. of Electrical Engineering, Chang Gung University, Taoyuan, "A Multilingual Automatic Speech Recognition (ASR) Engine Embedded on Personal Digital Assistant (PDA)" Cellular Neural Networks and Their Applications, 2005 9th International Workshop

[7] Common Health Risks of the Bedridden Patient Posted on October 24, 2013 by Carefect Blog Team

