VOICE COMMAND SYSTEM USING RASPBERRY PI

Surinder Kaur[1], Sanchit Sharma[2], Utkarsh Jain[3] and Arpit Raj[4]

Bharati Vidyapeeth's College of Engineering, New Delhi, India

ABSTRACT

The purpose of this research paper is to illustrate the implementation of a Voice Command System. This system works on the primary input of a user's voice. Using voice as an input, we were able to convert it to text using a speech to text engine. The text hence produced was used for query processing and fetching relevant information. When the information was fetched, it was then converted to speech using speech to text conversion and the relevant output to the user was given. Additionally, some extra modules were also implemented which worked on the concept of keyword matching. These included telling time, weather and notification from social applications.

KEYWORDS

Text to speech; Speech to text; Raspberry Pi; Voice Command System; Query Processing

1. INTRODUCTION

A Voice Command System essentially means a system that processes voice as an input, decodes or understands the meaning of that input processes it and generates an appropriate voice output. Any voice command system need three basic components which are speech to text converter, query processor and a text to speech converter. Voice has been a very integral part of communication nowadays. Since, it is faster to process sound and voices than to process written text, hence voice command systems are omnipresent in computer devices.

There have been some very good innovations in the field of speech recognition. Some of the latest innovations have been due to the improvements and high usage of big data and deep learning in this field [1]. These innovations have attributed to the technology industry using deep learning methods in making and using some of the speech recognition systems. Using big data for speech systems, Google was able to reduce word error rate by 6% to 10% relative, for the system that had the word error rate of 17% to 52% [2].

Text to speech conversion is the process of converting a machine recognized text into any language which could be identified by a speaker when the text is read out loud. It is two step processes which is divided into front end and back end [3]. First part is responsible for converting numbers and abbreviations to a written word format. This is also referred to as normalization of text. Second part involves the signal to be processed into an understandable one [4].

Speech Recognition is the ability of machine for instance a computer to understand words and sentences spoken in any language [5]. These words or sentences are then converted to a format that could be understood by the machine. Speech recognition is basically implemented using vocabulary systems [6]. A speech recognition system may be a Small Vocabulary-many user system or a Large Vocabulary- small user system [7].

DOI:10.5121/acii.2016.3306

2. SYSTEM ARCHITECTURE

2.1. EXISTING SYSTEM

The existing system suffers from the drawback that only predefined voices are possible and it can store only limited voices. Hence, the user can't get the full information coherently.

2.2. PROPOSED SYSTEM

The proposed system is such that it can overcome the drawback of the existing system. The project design involve text to speech. Here whatever the system receives as input after the command the output will get in the form of voice means speech.

2.3. HARDWARE IMPLEMENTATION

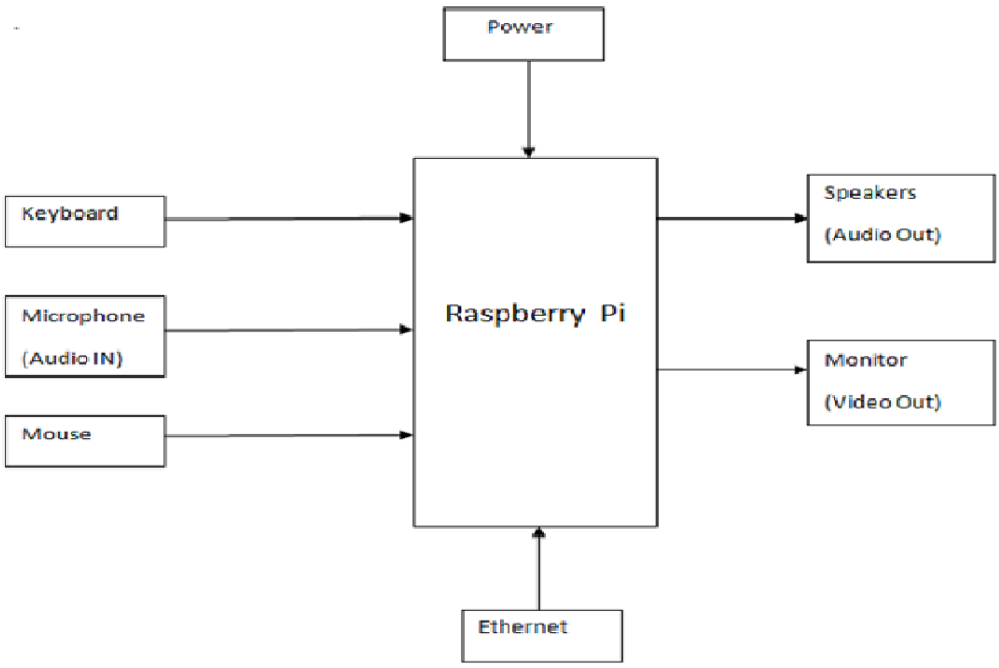

Figure 1.Hardware setup of Voice Command System

2.3.1. MICROPHONE

Microphone is used to take the audio input of the sound. This audio input when further passed through the system would be searched for keywords. These keywords are essential for the functioning of the voice command system as our modules work on the essence of searching for keywords and giving output by matching keywords.

2.3.2. KEYBOARD

Keyboard acts as an input interface mainly for the developers, providing access to make edits to the program code.

2.3.3. MOUSE

Mouse also acts an interface between the system and the developer and does not have a direct interaction with the end user.

2.3.4. RASPBERRY PI

It is the heart of the voice command system as it is involved in every step of processing data to connecting components together [8]. The Raspbian OS is mounted onto the SD card which is then loaded in the card slot to provide a functioning operating system.

2.3.5. POWER

The Raspberry Pi needs a constant 5V, 1.2 mA power supply. This can either be provided through an AC supply using a micro USB charger or through a power bank.

2.3.6. ETHERNET

Ethernet is being used to provide internet connection to the voice command system. Since the system relies on online text to speech conversion, online query processing and online speech to text conversion hence we need a constant connection to achieve all this.

2.3.7. MONITOR

Monitor provides the developer an additional way to look at the code and make any edits if any. It is not required for any sort of communication with the end user.

2.3.8. SPEAKERS

Once the query put forward by the user has been processed, the text output of that query is converted to speech using the online text to speech converter. Now this speech which is the audio output is sent to the user using the speakers which are running on audio out.

2.4. FLOW OF EVENTS IN VOICE COMMAND SYSTEM

First, when the user starts the system, he uses a microphone to send in the input. Basically, what it does is that it takes sound input from the user and it is fed to the computer to process it further. Then, that sound input if fed to the speech to text converter, which converts audio input to text output which is recognizable by the computer and can also be processed by it.

Then that text is parsed and searched for keywords. Our voice command system is built around the system of keywords where it searches the text for key words to match. And once key words are matched then it gives the relevant output.

This output is in the form of text. This is then converted to speech output using a text to speech converter which involves using an optical character recognition system. OCR categorizes and identifies the text and then the text to speech engine converts it to the audio output. This output is transmitted via the speakers which are connected to the audio jack of the raspberry pi.

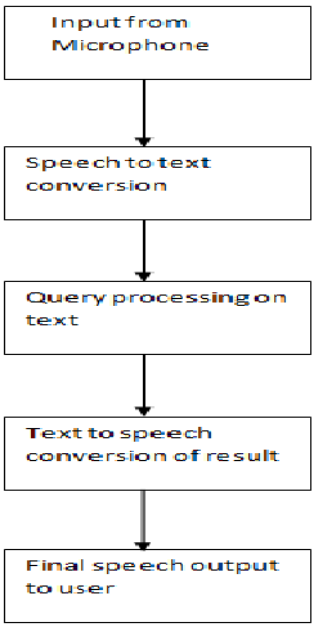

Figure 2. Flow of Events in Voice command System

3. MODULES IMPLEMENTED

3.1. SPEECH TO TEXT ENGINE

The Google speech engine is a Speech-To-Text (STT) engine which is used to convert the commands given by the user in audio input to text form, so that these commands can be interpreted by the modules properly. To use the Google speech engine, an application has to be created in the Google developers console and the generated API key has to be used to access the speech engine. It requires continuous internet connection as data is sent over the Google servers.

3.2. TEXT TO SPEECH ENGINE

CMU Flite (festival-lite) is a small, fast run-time text to speech synthesis engine developed at CMU and primarily designed for small embedded machines and/or large servers [9]. Flite is designed as an alternative text to speech synthesis engine to Festival for voices built using the FestVoxsuite of voice building tools. It is an offline engine and thus internet is not required to convert the text responses to audio responses. It is one of the fastest engines available for use [10].

3.3. QUERY PROCESSOR

The Voice Command System has a module for query processing which works in general like many query processors do. That means, taking the input from the users, searching for relevant outputs and then presenting the user with the appropriate output. In this system we are using the site wolfram alpha as the source for implementing query processing in the system. The queries that can be passed to this module include retrieving information about famous personalities, simple mathematical calculations, description of any general object etc.

3.4. FIND MY IPHONE

Find my iPhone module lets the users find their iPhone by their voice. This is done using the user's iCloud id and password to make the connection to the server. After the authentication of the password it checks the status of the iPhone. After checking, it connects to iPhone using the iCloud. After the connection, the command for ringing is sent to the iPhone to start a sound and notification on the phone for detection. If a failure occurs, the system generates an error that the iPhone is not found. An error is also generated when there are multiple iPhones associated with a single apple id.

3.5. MOVIES

The voice command system searches for relevant movie by the keyword "movie". It is implemented using a function which asks the user, the movie they want to know more about. Then a function is called to ask for the movie the user needs to know about. It searches for the top five results regarding the movie name. It confirms the movie among the listed names. On confirmation of the movie, it gives the detailed information including the rating, producer, director, cast, runtime and genres. In case of the failure, it generates the error of "Unable to find information on the requested movie".

3.6. WIKIPEDIA

This module works on the keyword of "wiki". The system asks for what you would like to learn about. Then the request is made to the Wikipedia API for the required query. It generates the summary of the information regarding the query and the data is output through the microphone to the listener in audio form. In case of failure, the error message is generated saying "unable to reach dictionary of wiki".

3.7. WEATHER

This module tells the user about the weather conditions of the location whose station identifier is specified in the profile of the user. This module can be executed by using the keyword "weather". The weather information is taken from the weather underground service which includes the details of temperature, wind speed and direction etc. It generates an error message, if the information cannot be retrieved for the specified location.

3.8. NEWS

News module can be executed by using the keyword "news". The headlines of top articles are retrieved from the internet using google news. The system tells the user all these headlines and asks the user if any of these articles should be sent to the user's email address. If the user specifies the number of the article to be sent, the article is sent to the specified email address. Otherwise, no further action is taken. If any failure occurs in retrieving headlines or sending articles, a corresponding error message is generated.

3.9. DEFINE

Define module is used to fetch the definitions of word specified by the user.The execution of this module is started by the keyword "define". Then, the user is asked for the word whose definition has to be provided. The module checks for the definitions of the specified word using the Yandex Dictionary API. All the available definitions are provided as a response from the API and the

same is given as output to the user in audio form. If the connection to api can not be established properly, then the error message "Unable to reach dictionary API" is generated for the user.

3.10. STATUS

This module can be used to retrieve the details of the status of the current system under execution by using the keyword "status". The information like current operating system version, running CPU percentage, number of CPU cores, system name and current memory utilization is checked by using the python system and process utilities and the output is given to the user in audio form.

3.11. JOKE

Joke module can be used for entertainment purposes by the user. This module works on the keywords "joke" or "knock knock". The jokes used in this module are predefined in a text file from which the jokes are read in a random order. A start and end line is present in every joke to differentiate it from others present in the file. All the lines of a joke are spoken by the system in the specified order only.

3.12. UNCLEAR

This module is used to generate an error message if the keyword specified by the user does not match the keywords present in any module of the system. This module has an empty array of keywords and is allocated the least priority, so that it is executed only after the keyword has been matched with all the other modules of the system. The messages generated by this module include statements like "I beg your pardon", "Say that again", "I'm sorry, Could you repeat that" and "My apologies, Could you try saying that again" which are selected in a random order each time this module is executed.

3.13. OTHER COMMAND SPECIFIC MODULES

The Voice Command System also has some command specific modules like fetching hacker news, email and current time. Each of these modules is related to the system using keywords like "hacker news", "email" and "time" respectively. Whenever any of this keyword is said to the system, it fetches that module and launches the contents of that module thereby providing the appropriate response to the user.

4. RESULT

The Voice Command System works on the idea and the logic it was designed with. Our system uses the keyword ***"listen"*** to take a command. Each of the commands given to it is matched with the names of the modules written in the program code. If the name of the command matches with any set of keywords, then those set of actions are performed by the Voice Command System. The modules of Find my iPhone, Wikipedia and Movies are based upon API calling. We have used open source text to speech and speech to text converters which provide us the features of customizability. If the system is unable to match any of the said commands with the provided keywords for each command, then the system apologizes for not able to perform the said task. All in all, the system works on the expected lines with all the features that were initially proposed. Additionally, the system also provides enough promise for the future as it is highly customizable and new modules can be added any time without disturbing the working of current modules.

5. CONCLUSIONS

In this paper, we have successfully given the idea and rationale behind the Voice Command System. We have also explained the flaws in the current system and our way of resolving those flaws. Additionally, we have also laid out the system architecture of our Voice Command System. Many of our modules are of open source systems and we have customized those modules according to our system. This helps get the best performance from the system in terms of space time complexity.

The Voice Command System has an enormous scope in the future. Already, we are seeing virtual assistants like Siri, Google Now and Cortana become popular in the mobile industry. This makes the transition smooth to a complete voice command system. Additionally, this also paves way for a Connected Home using Internet of Things and the voice command system.

REFERENCES

[1] Dahl, George E., et al. "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition." Audio, Speech, and Language Processing, IEEE Transactions on 20.1 (2012): 30-42.

[2] Chelba, Ciprian, et al. "Large scale language modeling in automatic speech recognition." arXiv preprint arXiv:1210.8440 (2012).

[3] Schultz, Tanja, Ngoc Thang Vu, and Tim Schlippe. "GlobalPhone: A multilingual text & speech database in 20 languages." Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on. IEEE, 2013.

[4] Tokuda, Keiichi, et al. "Speech synthesis based on hidden Markov models."Proceedings of the IEEE 101.5 (2013): 1234-1252.

[5] Singh, Bhupinder, Neha Kapur, and Puneet Kaur. "Speech recognition with hidden Markov model: a review." International Journal of Advanced Research in Computer Science and Software Engineering 2.3 (2012).

[6] Lamere, Paul, et al. "The CMU SPHINX-4 speech recognition system." IEEE Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP 2003), Hong Kong. Vol. 1. 2003.

[7] Black, Alan W., and Kevin A. Lenzo. "Flite: a small fast run-time synthesis engine." 4th ISCA Tutorial and Research Workshop (ITRW) on Speech Synthesis. 2001.

[8] Lee, Chin-Hui, Frank K. Soong, and Kuldip Paliwal, eds. Automatic speech and speaker recognition: advanced topics. Vol. 355. Springer Science & Business Media, 2012.

[9] Junqua, Jean-Claude, and Jean-Paul Haton. Robustness in automatic speech recognition: Fundamentals and applications. Vol. 341. Springer Science & Business Media, 2012.

[10] “Raspberry Pi Model B” [Online].Available: https://www.raspberrypi.org/products/model-b/. [Accessed November, 12, 2015]