# Speech Recognition System For Upper-Limb Impairment: A Text Editor For Programming via Voice

**Dharshan R[1], Santhosh S[2] and Amala Dejoe Nithin J[3]**

**Abstract**

Programming is arguably one of the most important skills to learn for the present and future generations. However, those with hand disabilities may find it difficult to program with a traditional keyboard and mouse. Disabled people would greatly benefit from assistive technology that allows them to use their voices to control computers. The existing system recognizes user input as voice and translates it into a program. However, the existing system only recognizes a limited number of keywords in a programming language. It would also be difficult to debug errors if the voice was misinterpreted. Our suggested work in speech recognition using natural language processing (NLP) would perform better than the previously available systems. Developing tools with a publicly available voice recognition API can serve as a foundation for programming without a keyboard. Voice control of the user interface for program execution and code debugging would also be quite beneficial. This can be done with the help of a Python modules: Pynput. Keyboard, PyAutoGUI, which can simulate mouse cursor movements and clicks as well as keyboard button presses. The accuracy of our proposed system is 87%. When converting voice commands to text, the Word Error Rate (WER) is 18%.

**Keywords**

Speech Recognition, Speech-to-Text, Vocal Programming, Python

## Introduction

Natural language processing (NLP) refers to the branch of computer science and more specifically, the branch of artificial intelligence or AI concerned with giving computers the ability to understand the text and spoken words in much the same way human beings can. It identifies and interprets words and phrases in spoken language and converts them into texts by computers. Natural Language Processing simply deals with the interaction between humans and computers using a natural language such as English. Natural Language Processing technology applies machine-learning algorithms to text and speech. Nowadays, the world depends on automation thus; computer programming is an essential one to learn. People with upper-limb dysfunction, on the other hand, have a hard time learning to program on computers. Our proposal to solve these issues has been mentioned in this paper.

## Problem Statement

Programming is primarily reliant on traditional keyboards for text entry and mouse control of user interfaces. Accessing computers to study programming is challenging for those with upper-limb dysfunction (or) without hands.

## Literature Survey

L Ashok Kumar in [1] proposes an assistive technology for hearing impaired people by using Audio Visual Speech Recognition (AVSR) technique. Trained RNN-GRU based speech to text model and Convolutional Neural Network based visual speech to text model. Fusion of audio visual speech recognition technique. Lowered word error rate of about 6.59% for ASR system and accuracy of about 95% using lip reading model. Word Error Rate (WER) of this proposed model need to be improved (can be achieved using BERT based language model). Bártek in [2] proposes a WebGen system to enable the creation of web pages by means of dialogue (utilizing the methods used in the wizards). Enables visually impaired 12 users creating the website step-by-step by some set of predefined web page templates. Only few predefined web page templates. Even though this proposal is new and futuristic, it would to be difficult for visually impaired users to access this system without any assistance. Joshi in [3] proposes a model where the main tasks are database access, speech-to-text conversion and to execute the corresponding commands. Database Access : Embedded database SQLite was used. Speech-to-text conversion : Time required to do the actual Speech to text conversion is user dependent. The conversion time is directly proportional to the time user speaks. Command Execution : After the recognition engine parse the command, the model will check the corresponding command in the database and orders the OS to execute it. If a user speaks fast then the speech recognition engine will not understand. Comparing it with already available project 'dragon' in the market, for single word conversion it performs less effective. Surahio in [4] proposed a model based on the praxis Hidden Markov Model. The spoken words are matched with the database and it puts on the document. It gives voice input for Five Language models : Dictionary, HTML, PHP, Special Characters, IDE for command and control. Hidden Markov model (HMM) is inappropriate

[1]Madras Institute of Technology
[2]Madras Institute of Technology
[3]Madras Institute of Technology

for short time intervals. Only limited number of keywords are recognized in each Language. G. Parthasarathy in [5] proposed a system where users can code just by using their voice. Used houndify SDK which matches the spoken word with most accurate word and responses with json format which is then processed and converted into suitable python code and displayed in trinket IDE. Used pyautogui python module to control screen automations. Proposed only for python programming language. Accuracy need to be improved. Maloku in [6] proposed a model to program in Java Language through voice. Technologies used: Dragon NaturallySpeaking - Speech Recognition Software, NatLink - Speech Recognition API, DragonFly - Python library. Speech Recognition takes more time than using traditional Keyboard and Mouse. Proposed only for JAVA programming language.

## Algorithms and Models

### A. Hidden Markov Model

Hidden Markov Model is the base of a set of successful techniques for acoustic modeling in speech recognition systems. The main reasons for this success are due to this model's analytic ability in the speech phenomenon. The HMM model parameters represent the time-varying characteristics of the voice signal. It consists of two interrelated stochastic processes common to describe the statistical characteristics of the signal. One of which is hidden (unobserved) finite state Markov chain, and the other is the observation vector associated with each state of the Markov chain stochastic process (observable). Reveal characteristics of the hidden Markov chain depends on the signal characteristics can be observed. In this way, a certain period of time varying signals such as voice characteristics described by the random process corresponding to the symbols of state observation. Signal described by the hidden Markov chain transition probability chances with time.

### B. Dragon Naturally Speaking

Dragon Naturally Speaking is speech recognition software. Dragon uses voice commands to dictate, edit and control computer/mobile device 16 applications. Dragon is used for physical disabilities, learning disabilities, and medical disabilities. Dragon Naturally Speaking is a dictation software that helps students and professionals alike. While initially it takes a bit of time to get started using the software proficiently, Dragon will save its user a lot of time. The students in our study learned the software in anywhere from a day or two to a month, with most getting the hang of it in a few weeks. While Dragon gets trained to a single voice per user, multiple users are allowed; however, each user must train independently from one another. Dragon shines as a tool for writing papers, or other homework that involves notetaking. Dragon users can also craft emails using the same word processing technology.

### C. Audio-visual speech recognition

Audio-visual speech recognition (AVSR) is a technique that uses image-processing capabilities in lip reading to aid speech recognition systems in recognizing nondeterministic phones or giving preponderance among near probability decisions. Each system of lip reading and speech recognition works separately, then their results are mixed at the stage of feature fusion. As the name suggests, it has two parts. First one is the audio part and second one is the visual part. In audio part we use features like log mel spectogram, mfcc etc. from the raw audio samples and we build a model to get feature vector out of it . For visual part generally we use some variant of convolutional neural network to compress the image to a feature vector after that we concatenate these two vectors (audio and visual ) and try to predict the target object.

### D. Houndify SDK for Speech Recognition

Houndify uses the collective intelligence technique which is an independent voice AI platform, that allows developers to build custom voice assistants with a branded wake word for any product or service. Speech-to-Text engine delivers unprecedented speed and accuracy.

### Disadvantages of the Existing models

1. Debugging of code through voice is difficult
2. Lacks accuracy and misinterpretation
3. Background noise interference

## Proposed Work

Python is an emerging language in recent times, so we decided to develop a text editor for python, which accepts input through voice and converts it into an equivalent python code.

### Speech Recognition Engine

Speech Recognition incorporates computer science and linguistics to identify spoken words and converts them into text. This is done with the help of Google Speech Recognition. This requires an active internet connection to work. The Google Speech API enables developers to convert audio to text by applying powerful neural network models. Speech recognition in python with the help of computer programs that takes speech as input from the microphone, process it, and convert it into an equivalent text.

### Switching Between Modes

Voice note is then checked to choose the modes. If it is a coding mode then the voice is processed and converted into equivalent python code. If it is a command mode then terminal commands like save, run, download are executed.

1. Command mode: In this mode, the obtained text is matched with the system commands such as run, save, open, etc.
2. Coding mode: In this mode, the obtained text is matched with the python syntax templates. Using these modes, we can able to interact with the text editor and run the programs. This method is even more easy to use than the previously existing text editor.

Voice control of the user interface for program execution and code debugging can be done with the help of a Python module called PyAutoGUI, which can simulate mouse cursor movements and clicks as well as keyboard button presses. Our design starts with the voice input and ends with the python source code and its execution. In Fig-1, the microphone icon shows the voice input and it was fed into the speech recognition engine where the audio processing, analysis and speech-to-text functionalities takes place. Then, the generated text enters the keyword identification section.

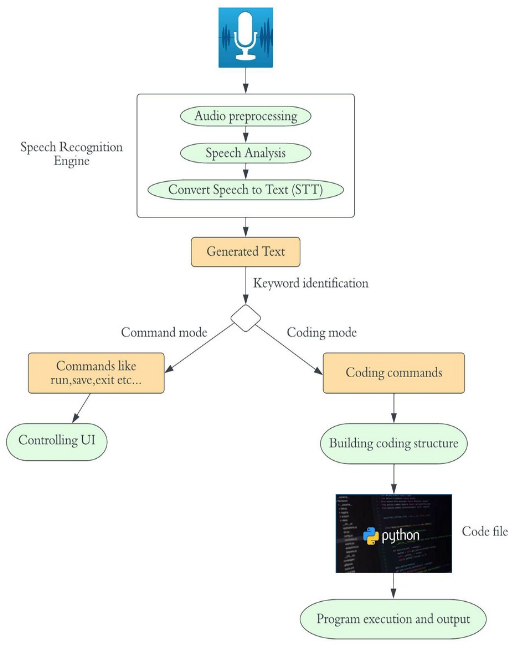

Audio preprocessing
Speech Analysis
Convert Speech to Text (STT)
Speech Recognition Engine
Generated Text
Keyword identification
Command mode
Coding mode
Commands like run,save,exit etc...
Coding commands
Controlling UI
Building coding structure
python
Code file
Program execution and output

*Fig-1. System architecture*

If the text matches with the controlling commands, it will do the respective UI control Functions, else if it matches with coding commands, it will insert the right syntax. Otherwise it will insert the generated texts.

## Tools and Libraries Used

### *Python*

Python is an interpreted, interactive, object-oriented programming language. It incorporates modules, exceptions, dynamic typing, very high level dynamic data types, and classes. It supports multiple programming paradigms beyond object-oriented programming, such as procedural and functional programming.

### *Speech Recognition Library*

Library for performing speech recognition, with support for several engines and APIs, online and offline.

### *Google Speech API*

Google Speech API enables developers to convert audio to text by applying powerful neural network models in an easy to use API. Speech to text translation(STT) is done with the help of Google Speech Recognition. This requires an active internet connection to work. Google Speech Recognition is one of the easiest to use. It returns text results in real-time.It is accurate in noisy environments.It works with apps across any device and platform.

### *Pywinauto*

pywinauto is a set of python modules to automate the Microsoft Windows GUI. At its simplest it allows you to send mouse and keyboard actions to windows dialogs and controls.

### *Pyautogui*

PyAutoGUI is a cross-platform GUI automation Python module for human beings. Used to programmatically control the mouse  keyboard. Sending keystrokes to applications (for example, to fill out forms). Take screenshots, and given an image (for example, of a button or checkbox), and find it on the screen. Locate an application's window, and move, resize, maximize, minimize, or close it (Windows-only, currently). Display alert and message boxes.

### *Pynput*

This library allows you to control and monitor input devices. Currently, mouse and keyboard input and monitoring are supported.

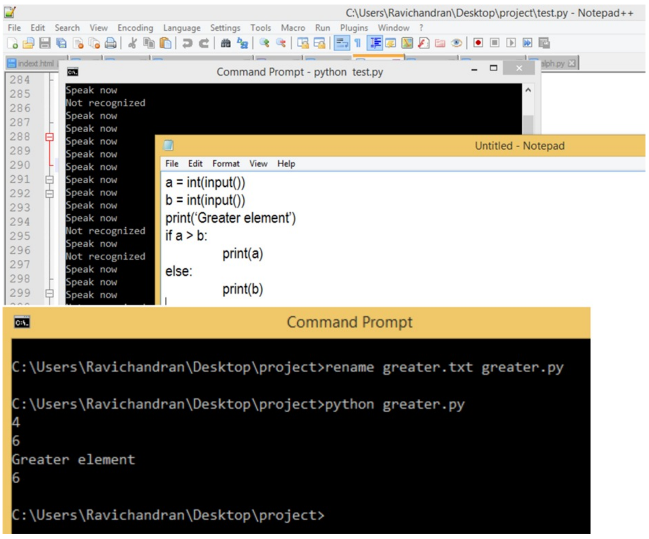
**Algorithm for converting voice to Python code**

1. Start
2. Importing the necessary libraries : SpeechRecognition, pywinauto, pyautogui.
3. Run the python program from command prompt.
4. Notepad application will be automatically woken with the help of application module, which is the sub-module of pywinauto module.
5. Input is collected through the microphone.
6. Voice note is then checked to choose the modes. If it is a coding mode then the voice is processed and converted into equivalent python code. If it is a command mode then terminal commands like save, run, download are executed.
   (a) Command mode: In this mode, the obtained text is matched with the system commands such as run, save, open, etc.
   (b) Coding mode: In this mode, the obtained text is matched with the python syntax templates. Using these modes, we can able to interact with the text editor and run the programs. This method is even more easy to use than the previously existing text editor (or IDEs).
7. Voice control of the user interface for program execution and code debugging can be done with the help of a Python modules,
   (a) PyAutoGUI is used to programmatically control the mouse  keyboard.
   (b) Pynput : This library allows you to control and monitor input devices. Currently, mouse and keyboard input and monitoring are supported.
8. The program is executed and the output of the program is displayed on new command prompt.
9. End

## Implementation and Result

The below given snapshots are some of the implementation examples of the proposed system.

### *Python program to find the greater element*


C:\Users\Ravichandran\Desktop\project\test.py - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Command Prompt - python test.py
Untitled - Notepad
File Edit Format View Help
Command Prompt

```
Speak now
Not recognized
Speak now
Speak now
Speak now
Speak now
Speak now
Speak now
Speak now
Speak now
Speak now
Not recognized
Speak now
Not recognized
Speak now
Speak now
Speak now
```

```
a = int(input())
b = int(input())
print('Greater element')
if a > b:
        print(a)
else:
        print(b)
```

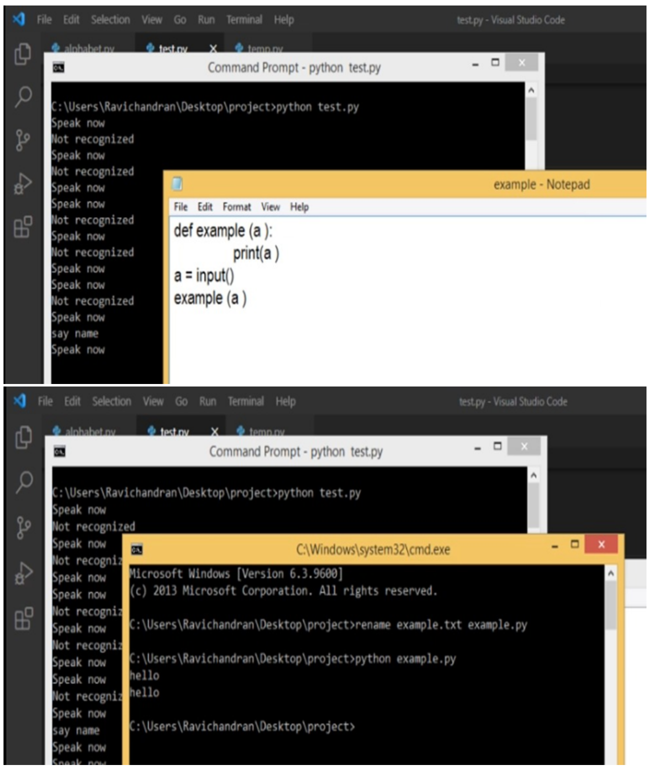
```
C:\Users\Ravichandran\Desktop\project>rename greater.txt greater.py

C:\Users\Ravichandran\Desktop\project>python greater.py
4
6
Greater element
6

C:\Users\Ravichandran\Desktop\project>
```

*Fig-2. greater element - example*

### *Python program - functions*


File Edit Selection View Go Run Terminal Help
test.py - Visual Studio Code
alphabet.py
test.py
temp.py
Command Prompt - python test.py
example - Notepad
File Edit Format View Help
C:\Windows\system32\cmd.exe

```
C:\Users\Ravichandran\Desktop\project>python test.py
Speak now
Not recognized
Speak now
Not recognized
Speak now
Speak now
Not recognized
Speak now
Not recognized
Speak now
Speak now
Not recognized
Speak now
say name
Speak now
```

```
def example (a ):
        print(a )
a = input()
example (a )
```

```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\Ravichandran\Desktop\project>rename example.txt example.py

C:\Users\Ravichandran\Desktop\project>python example.py
hello
hello

C:\Users\Ravichandran\Desktop\project>
```

*Fig-3. functions - example*

### *Accuracy Test - Command Mode*

Command mode table describes the test and result data of the commands in our project. The maximum error observed for the single command was 2. The program gives more than 88% of accuracy level for the command mode. The average response time for this command mode was 5s. This mode performs better than the existing projects.

| Train Data | Number of Test | Number of correct word | Error | Average Response Time |
|---|---|---|---|---|
| back (or) delete | 10 | 9 | 1 | 5.9s |
| next line | 10 | 8 | 2 | 6.3s |
| select top<br>select bottom<br>select right<br>select left | 10 | 8 | 2 | 6.1s |
| Top<br>bottom<br>left<br>right | 10 | 9 | 1 | 5.8s |
| Cut | 10 | 9 | 1 | 5.6s |
| Copy | 10 | 10 | 0 | 6.0s |
| Paste | 10 | 10 | 0 | 5.9s |
| select all | 10 | 10 | 0 | 4.5s |
| Tab | 10 | 9 | 1 | 5.8s |
| Spacebar | 10 | 8 | 2 | 6.1s |
| Backspace | 10 | 9 | 1 | 6.3s |
| Undo | 10 | 8 | 2 | 5.8s |
| Intensify | 10 | 8 | 2 | 6.8s |
| save the program | 10 | 9 | 1 | 6.9s |
| Open | 10 | 8 | 2 | 5.9s |
| Execute | 10 | 9 | 1 | 6.1s |
| Exit | 10 | 9 | 1 | 5.6s |
| Close | 10 | 10 | 0 | 5.4s |

*Table-1. Accuracy Test - Command Mode*

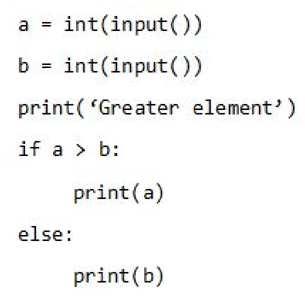
### *Accuracy Test - Coding Mode*

Coding mode table describes the test and result data of the coding syntax commands in our project. The maximum error observed for the single command was 3. The program gives more than 87% of accuracy level for the command mode. The average response time for this command mode was 6s. This coding mode performs better than the existing projects. The maximum response time observed was 12s for the class command.

| Train Data | Number of Test | Number of correct word | Error | Average Response Time |
|---|---|---|---|---|
| + - * / % ^ | 10 | 10 | 0 | 5.9s |
| = == != | 10 | 9 | 1 | 6.2s |
| and, or | 10 | 8 | 2 | 5.5s |
| <> | 10 | 9 | 1 | 5.7s |
| : ; , | 10 | 9 | 1 | 5.5s |
| ‘ ’, “ ” | 10 | 9 | 1 | 6.3s |
| ( ) { } [ ] | 10 | 10 | 0 | 6.7s |
| Print | 10 | 9 | 1 | 6.5s |
| user input | 10 | 9 | 1 | 6.4s |
| Casting | 10 | 7 | 3 | 6.0s |
| Characters | 10 | 9 | 1 | 6.9s |
| Numbers | 10 | 9 | 1 | 6.7s |
| if – else | 10 | 10 | 0 | 6.4s |
| for loop | 10 | 8 | 2 | 6.5s |
| while loop | 10 | 8 | 2 | 6.3s |
| Functions | 10 | 8 | 2 | 6.7s |
| class and object | 10 | 8 | 2 | 12.9s |
| inserting command words | 10 | 9 | 1 | 5.8s |

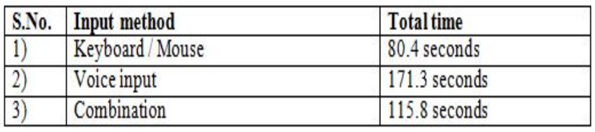
*Table-2. Accuracy Test - Coding Mode*

**Example Program-1 :-**



```
a = int(input())
b = int(input())
print('Greater element')
if a > b:
     print(a)
else:
     print(b)
```

This table shows the total time taken to code the Example Program-1 using different input methods.



| S.No. | Input method | Total time |
|---|---|---|
| 1) | Keyboard / Mouse | 80.4 seconds |
| 2) | Voice input | 171.3 seconds |
| 3) | Combination | 115.8 seconds |

*Table-3. Total time for all tasks separated in sections*

Here, we are testing our system with the simple 'greater element' python program. Analysing the time required for the tasks to be done by the different input methods and tabulating the time taken by different inputs.

## Conclusion and Future Work

Our proposed model supports vocal programming and demonstrates that it is feasible to use the model for programming. We contributed a simple voice-to-code program, as well as a study of its usability, potential, and accuracy for individuals who have upper-limb impairments. It could make programming possible for anyone, regardless of their physical ability. With the help of the Google Speech API, voice commands are recognised (Word Error Rate (WER) : -18%). The overall accuracy of our proposed model is 87%. In our future work, we plan to improve this accuracy by proper implementation. This model clearly shows that anyone can code without difficulty using this model. We are also planning to release this model for other languages like C, C++, Java, etc. in the near future.

## References

[1] L Ashok Kumar, D Karthika Renuka, S Lovelyn Rose, M C Shunmuga priya, I Made Wartana, "Deep learning based assistive technology on audio visual speech recognition for hearing impaired", International Journal of Cognitive Computing in Engineering(2022), DOI:https://doi.org/10.1016/j.ijcce.2022.01.003.

[2] Bártek, Luděk Plhák, Jaromír, "WebGen System - Visually Impaired Users Create Web Pages", International Conference on Computers for Handicapped (ICCHP) (2008), DOI: 10.1007/978-3-540-70540-6_67.

[3] Joshi, Pranali Patki, Ravi, "Implementation of Voice User Interface using Speech Recognition", International Journal of Computer Applications (2015), DOI:124. 37-42. 10.5120/ijca2015905633.

[4] Surahio, Farhan Ali Jumani, Awais Talpur, "An Approach to accept input in Text Editor through voice and its Analysis, designing, development and implementation using Speech Recognition", International journal of computer science and network security (2016).

[5] G. Parthasarathy, A R. Rangeesh, V S. Sri Kishowr, R. Sriram, S. Vijay, "An Approach to Accept Voice in Code Editor through Speech Recognition", INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH TECHNOLOGY (IJERT) ICONNECT – 2018.

[6] Maloku, Rinor Pllana, Besart, "HyperCode: Voice aided programming", IFAC (2016), DOI : 10.1016/j.ifacol.2016.11.073.

[7] H. Chadha, S. Mhatre, U. Ganatra and S. Pathak, "HTML Voice," 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), 2018, pp. 1-4, DOI: 10.1109.

[8] A.S. Spanias, "Speech coding: a tutorial review," in Proceedings of the IEEE, vol. 82, no. 10, DOI: 10.1109.

[9] J. D. Gibson, "Speech coding methods, standards, and applications," in IEEE Circuits and Systems Magazine, vol. 5, no. 4, pp. 30-49, Fourth Quarter 2005, DOI: 10.1109/MCAS.2005.1550167.

[10] Arnold, S. C., Mark, L., Goldthwaite, J., Programming by voice, VocalProgramming. In Proceedings of the fourth international ACM conference on Assistive technologies (pp. 149-155)- 2000 ACM.

[11] Begel, A., Graham, S. L. (2006). An assessment of a speech-based programming environment. In Visual Languages and Human-Centric Computing, 2006. VL/HCC 2006. IEEE Symposium on (pp. 116-120). IEEE.