

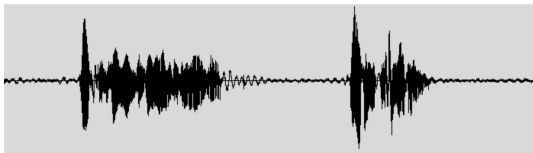
Interact with Word Document Using Cursor and Speech Recognition

Presented by: Hang Zhao

Stony Brook University

December 2, 2019

Introduction



► Speech Recognition

1. Coverts human language to computer readable content
2. Roots in earlier 1950s at Bell Labs
3. Being commonly used among people's life such as Amazon Alexa and Apples Siri.
4. Allow the elderly and the physically and visually impaired person to interact with the world quickly and naturally.

Introduction

- ▶ Using cursor and speech recognition to control document
- ▶ Limits of traditionally typing
 1. Be able to pronounce a word but fail to spell
 2. In a situation not convenient to type
- ▶ Recent work are mostly focus on improving the correctness of speech recognition by training all kinds of dataset
- ▶ Another one is using speech recognition to control cursor

Introduction

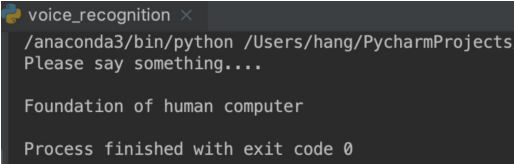
```
import speech_recognition as sr

r = sr.Recognizer()
mic = sr.Microphone()

with mic as source:
    r.adjust_for_ambient_noise(source)
    print("Please say something...\n")
    audio = r.listen(source)
    print(r.recognize_google(audio))
```

- ▶ This project is aiming to combine cursor and speech recognition to interact with word document by using several python libraries together
 1. Tkinter is a standard pythonGUI API which is used for interacting with users
 2. Speech recognition provides APIs such as Microsoft Bing Speech, GoogleWeb Speech API, CMU Sphinx etc. Each of them have a class named Recognizer which includes many built-in function to handle audio source

Introduction

A terminal window titled 'voice_recognition' with a close button. The command executed is `/anaconda3/bin/python /Users/hang/PycharmProjects/voice_recognition/main.py`. The output shows the prompt 'Please say something....', the text 'Foundation of human computer' entered, and the message 'Process finished with exit code 0'.

- ▶ 1. Pyautogui is a multi-platform GUI automatic library which uses programming to control cursor and keyboard
- 2. Pynput is a python library which is used for controlling the keyboard
- 3. Subprocess library provides the child process forked by the parent process to handle a particular task like executing open file
- 4. Docx document allows user to interact with word document by using python

Related Works

- ▶ Speech recognition can be dated back from 1950s. Three researchers there built up a single-speaker digit recognition system. This system can only accept single source speaker at a time and very limited words. After more than half century's development, speech recognition nowadays can accept multiple speakers at a time and can recognize a large amount of words among different languages. It depends on a famous model called Hidden Markov Model(HMM). Under this model, a continuous speech is defined as a time chain of states. There is a probability from one state to another state.

Related Works

- ▶ The speech signal is divided into 10-millisecond pieces in HMM. Each of these pieces is mapped to a vector of cepstral coefficients. And after implementing HMM, we can get a sequence of these vectors. These vectors are matched to a fundamental unit of speech call phonemes. Since phonemes differ from people to people, and even same people at different time can have different phonemes, this step requires a large data set to train the model. A lot of modern works among speech recognition are training the model.

Related Works

- ▶ Using windows speech recognition to click the mouse button. Bill Marcotte was using speech recognition to control mouse and interface. The example he gives is to close current window by simply speaking "Close Window". And another example is to use speech recognition to click, double-click and right click.

Methodology

- ▶ This project is using python tkinter library to embed several functions inside the buttons to achieve specific effects. In general, there are two ways to control the main menu: one is clicking button to achieve a particular function; another one is using speech recognition to control the command.

Methodology

Word Document Interactor



voice control	Using speech recognition to control each step!
open file	Speak : open file. Open word document from a directory!
save file	Speak : save file. Save word document to a directory!
create file	Speak : create file. Create word document in any directory!
make font bigger	Speak : make words bigger. Save word document to a directory!
make font smaller	Speak : make words smaller. Making selected words in document smaller!
underline	Speak : underline. Making selected words underline!
bold	Speak : make words bold. Making selected words bold!
italic	Speak : make words italic. Making selected words italic!
magic	Speak : magic. Changing the content to be the word recognized by speech!
exit	Speak : exit. Exit program!

Methodology

- ▶ Under the cover, there are several functions we want to achieve:
 1. The orange area shows the response of the system to the user
 2. Create a word document and save it in any directory
 3. Open a word document in any directory
 4. Save a word document in any directory
 5. Modify the word under the position of the cursor to be the word recognized by speech recognition
 6. Make the size of the word under the position of the cursor bigger
 7. Make the size of the word under the position of the cursor smaller
 8. Add underline to the word under the position of the cursor
 9. Make the word under the position of the cursor bold
 10. Make the word under the position of the cursor italic

Methodology

```
def modify_content():  
  
    r = sr.Recognizer()  
    mic = sr.Microphone()  
  
    with mic as source:  
        r.adjust_for_ambient_noise(source)  
        print("Please say something...\n")  
        text1.insert(INSERT, "Please say something...\n")  
        audio = r.listen(source)  
        ret = r.recognize_google(audio)  
        print("The word you said is : ", ret)  
        text1.insert(INSERT, "The word you said is : ", ret)  
        pyperclip.copy(ret)  
        time.sleep(2)  
  
    pyautogui.doubleClick()  
  
    keyboard = Controller()  
  
    keyboard.press(Key.cmd.value)  
    keyboard.press('v')  
    keyboard.release('v')  
    keyboard.release(Key.cmd.value)  
    pyautogui.typewrite(" ")
```

- ▶ Speech recognition API provides several function to allow users to interact with the voice and convert them into text. One of the most important class is called recognizer. This function can recognize the voice of a user. In the meantime, python PyAudio library offer a easy way to get the input source from microphone.

Methodology

- ▶ After we finish setting the microphone as source input, we can pass the value to the recognizer class. There is a function called `adjust_for_ambient_noise` which is used for preprocessing and reducing the noise level. Then we can let the microphone listen the user input. Next, we put the result into google recognizer which can only be invoked when the Internet is available. This step will result in converting voice into text.

Methodology

- ▶ At the same time, user should move the cursor to the word position at which he or she want to change and click the left button. Then the program will use pyperclip library to copy the content got from google recognizer. Then pyautogui library is used to double click this position. This would end up with selecting the whole word under the cursor position. Next, pynput library is invoked and keyboard controller is used by pressing command key and 'v' key together. The function of this is paste the content. Finally, those keys are released. This is the whole idea behind the word substitution.

Methodology

```
def create_file():  
    document = Document()  
    file_path = filedialog.asksaveasfilename(title=u'saving file')  
    print(file_path)  
    document.save(file_path + ".docx")
```

- ▶ Create file. This method is implemented by using python docx library and tkinter library. Docx provides a method document to create a word document. Meanwhile, tkinter pop out a new window to let the user choose a directory where the file will be saved. We can end up with getting the path name including the file name. At last, we save the file by invoking document save method.

Methodology

```
def open_file():  
    global file_path  
    global file_text  
    file_path = filedialog.askopenfilename(title=u'choose file',  
                                           initialdir=(os.path.expanduser('/User')))  
    print('open file: ', file_path)  
    if file_path is not None:  
        subprocess.run(['open', file_path], check=True)
```

- ▶ Open file. Tkinter pop out a new window to let the user choose a word document where the file will be opened. Then if the path is not empty, the main process will fork a child process to run this open file task in a new window.

Methodology

```
def save_file():
    global file_path
    global file_text
    file_path = filedialog.asksaveasfilename(title=u'save file')
    print('saving file: ', file_path)
    file_text = text1.get('1.0', tk.END)
    if file_path is not None:
        with open(file=file_path, mode='a+', encoding='utf-8') as file:
            file.write(file_text)
        text1.delete('1.0', tk.END)
        dialog.Dialog(None, {'title': 'File Modified',
                             'text': 'finish saving',
                             'bitmap': 'warning', 'default': 0,
                             'strings': {'OK', 'Cancel'}})
    print('finish saving file')
```

- Save file. Tkinter pop out a new window to let the user choose a directory where the file will be saved. Then the file will finish writing all the content into the file and save them when exit the file.

Methodology

```
def modify_font():  
    time.sleep(1)  
    pyautogui.doubleClick()  
    time.sleep(1)  
    keyboard = Controller()  
    with keyboard.pressed(Key.cmd.value):  
        with keyboard.pressed(Key.shift):  
            keyboard.press('.')  
    keyboard.release('.')  
    pyautogui.click()  
  
    keyboard.release(Key.shift)  
    keyboard.release(Key.cmd.value)
```

- Make font bigger. Using pyautogui double click method to choose the whole word under current cursor position. Then pynput is invoked to press command key, shift key and '.' key together to make the word selected bigger. This is the hotkey option, and pynput method here is just using hotkey to achieve this function. After achieving this function pyautogui single click is used to cancel selection and then command key, shift key and '.' key are released.

Methodology

```
def modify_font1():  
    time.sleep(1)  
    pyautogui.doubleClick()  
    time.sleep(1)  
    keyboard = Controller()  
    with keyboard.pressed(Key.cmd.value):  
        with keyboard.pressed(Key.shift):  
            keyboard.press(',')  
        keyboard.release(',')  
    pyautogui.click()  
  
    keyboard.release(Key.shift)  
    keyboard.release(Key.cmd.value)
```

- Make font smaller. Using pyautogui double click method to choose the whole word under current cursor position. Then pynput is invoked to press command key, shift key and ',' key together to make the word selected smaller. This is the hotkey option, and pynput method here is just using hotkey to achieve this function. After achieving this function pyautogui single click is used to cancel selection and then command key, shift key and ',' key are released.

Methodology

```
def underline():  
    time.sleep(1)  
    pyautogui.doubleClick()  
    time.sleep(1)  
    keyboard = Controller()  
  
    with keyboard.pressed(Key.cmd.value):  
        keyboard.press('u')  
        keyboard.release('u')  
        pyautogui.click()  
  
    keyboard.release(Key.cmd.value)
```

- Underline. Using pyautogui double click method to choose the whole word under current cursor position. Then pynput is invoked to press command key and 'u' key together to make the word selected underline. This is the hotkey option, and pynput method here is just using hotkey to achieve this function. After achieving this function pyautogui single click is used to cancel selection and then command key and 'u' key are released.

Methodology

```
def bold():  
    time.sleep(1)  
    pyautogui.doubleClick()  
    time.sleep(1)  
    keyboard = Controller()  
  
    with keyboard.pressed(Key.cmd.value):  
        keyboard.press('b')  
        keyboard.release('b')  
        pyautogui.click()  
  
    keyboard.release(Key.cmd.value)
```

- ▶ Make font bold. Using pyautogui double click method to choose the whole word under current cursor position. Then pynput is invoked to press command key and 'b' key together to make the word selected bold. This is the hotkey option, and pynput method here is just using hotkey to achieve this function. After achieving this function pyautogui single click is used to cancel selection and then command key and 'b' key are released.

Methodology

```
def Italic():  
    time.sleep(1)  
    pyautogui.doubleClick()  
    time.sleep(1)  
    keyboard = Controller()  
  
    with keyboard.pressed(Key.cmd.value):  
        keyboard.press('i')  
        keyboard.release('i')  
        pyautogui.click()  
  
    keyboard.release(Key.cmd.value)
```

- ▶ Make font Italic. Using pyautogui double click method to choose the whole word under current cursor position. Then pynput is invoked to press command key and 'i' key together to make the word selected italic. This is the hotkey option, and pynput method here is just using hotkey to achieve this function. After achieving this function pyautogui single click is used to cancel selection and then command key and 'i' key are released.

Methodology

Command	Function
open file	Open word document from a directory
save file	Save word document to a directory
create file	Create word document in any directory
make words bigger	Making selected words in document bigger
make words smaller	Making selected words in document smaller
underline	Making selected words in document underline
make words bold	Making selected words in document bold
make words italic	Making selected words in document italic
magic	Changing the content to be the word recognized by speech
exit	Exit program

- ▶ Voice control. Instead of clicking the button provided on the interface, we also provided user to use voice to control all the functions mentioned above. The accepting rules are listed above.

Experiments

- ▶ Due to the accurate rate of speech recognition, it will sometime end up getting an error of speech recognition of Unknown Value Error. This is because the microphone cannot get the content and hence we have to re-launch the program. But most of the time, voice matching accuracy is high enough to be accepted.

Conclusions

- ▶ Interacting with word document using cursor and speech recognition brings a lot convenience to people. And they are not necessary typing the word by themselves. Instead, using speech recognition and cursor to replace the word they want to change. Especially for the elderly and people who not good at spelling or memorizing words. The accurate rate of speech recognition nowadays can reach 95 percent or even higher. It is easier for computer to understand the human intention, and in the future there will be more speech recognition related products that come into human's life.

Reference

- ▶ Bakanay, H.(2019, November 1). *The Advantages of Speech Recognition Technology: SESTEK Blog*. Retrieved November 4, 2019, from <https://www.sestek.com/2014/11/the-advantages-of-speech-recognition-technology/>.
- ▶ Amos , D. (2018, June 28). *The Ultimate Guide To Speech Recognition With Python*. Retrieved November 4, 2019, from <https://realpython.com/python-speech-recognition/>.
- ▶ Pedro Cabalar, Sergei Odintsov and David Pearce. *Strong Negation in Well-Founded and Partial Stable Semantics for Logic Programs*. Springer, 2006.
- ▶ Huang, X., Baker, J., Reddy, R. (2014). *A Historical Perspective of Speech Recognition*. Communications of the ACM, 57(1), 94–103.

Reference

- ▶ Boyd, C. (2018, January 10). Retrieved from <https://medium.com/swlh/the-past-present-and-future-of-speech-recognition-technology-cf13c179aaf>.
- ▶ Pieraccini, R. (2012). *The voice in the machine: building computers that understand speech*. Cambridge, MA: MIT Press.
- ▶ Rabiner, L. R., Juang, B.-H. (2005). *Fundamentals of speech recognition*. Delhi: Pearson Education.
- ▶ Yu, D., Deng, L. (2016). *Automatic speech recognition: a deep learning approach*. London: Springer.
- ▶ Warakagoda, N. (1996, May 10). *Assumptions in the theory of HMMs*. Retrieved November 4, 2019, from <http://jedlik.phy.bme.hu/~gerjanos/HMM/node5.html>.

Reference

- ▶ Shmyrev, N. (n.d.). *Basic concepts of speech recognition*. Retrieved from <https://cmusphinx.github.io/wiki/tutorialconcepts/>.
- ▶ Bill Marcotte. (2011, April 21). *Using Windows Speech Recognition to click the mouse button*. Retrieved from <http://www.cameramouse.org/downloads/clicking>

Thank You