E-BOOK NAVIGATION BY VOICE COMMAND

SHRADHA HOLANI
2017MCS2105
Guide - Prof. M.Balakrishnan
Mentor - Akashdeep Bansal

1.Motivation

Screen readers like NVDA can be used to read an e-book using specific key-strokes which are predefined. This involves a lot of learning on user's end, as he needs to know all the key sequences for each command. For example to go to the next paragraph, he needs to press Ctrl+Down key sequence. Also on touch screen devices, for every action there is a specific gesture, introducing new gestures for other commands is limited by the number of possible simple gestures making things complicated. To reduce the learning overhead on users, and also avoiding the limited gesture space problem, using voice commands will give them complete independence to navigate through an e-book.

2. Literature Survey

Dictation Bridge

Dictation Bridge is an NVDA add-on..All the commands are added to the Speech Macros library, thus Windows Speech Recognition is able to recognize only those commands. This makes it inflexible and static.

I have studied Dictation Bridge, to understand how this works and if a similar idea can be used for our purpose. It's helpful if we're trying to develop our application as an add-on, in particularly the way it calls NVDA functions. I have also been discussing with the developer of Dictation Bridge, about these issues and trying out different ways to resolve it.

SendKeys

SendKeys class in C#, sends keystrokes to the active application. I tried using this class to send keyboard commands to the active screen reader(NVDA), and expected NVDA to trap those keys and perform the corresponding function. But till now, this is not able to execute in the desired manner, though the reason of it's failure is still not known.

Keyboard

Keyboard is a python library to control the keyboard. Some of the commands like “h”, “i”, “l” ,“Ctrl+home” etc are working, but commands like “up”, “ctrl+down” doesn’t work on browser.

PyAutoGUI

Pyautogui is a python module for programmatically controlling the keyboard,similar to keyboard module. Some of the commands like “h”, “Ctrl+home” etc are working her as well, but commands like “up”, “ctrl+down” doesn’t work on browser.

3. Steps Involved

- Translate voice command to text.
- Understand user’s intention behind the command.
- Execute the intended action.

3a. Convert Speech to Text

This can be done in both online and offline mode :

- Online: Google Speech Recognition API
  This involves calling the Google Speech Api. It takes voice as input and returns the corresponding recognized text. This gives a fairly good accuracy with a microphone.
  The required dependencies needs to be installed before using this service:
  - ❏ Python speech recognition module- This can be installed using the command pip install SpeechRecognition
  - ❏ Python PyAudio module- This can be installed using the command pip install pyaudio

The following code snippet converts audio, taking input from the microphone to text :

```
r = sr.Recognizer()
print("Speak Now!")
with sr.Microphone() as source:
      r.adjust_for_ambient_noise(source, duration = 1)
       audio = r.listen(source)
try:
      s=r.recognize_google(audio)  # recognize speech using Google Speech
Recognition
       print("You said " + s)
        return s                          #recognized text
except LookupError:                           # speech is unintelligible
        print("Could not understand audio")
```

- Offline: PocketSphinx
  This is a Java based library which works offline but has very low accuracy. Need to improve it's accuracy to be able to use it.

3b. Understanding User's intent

Previous Approaches Tried:

- Created a dictionary with all the possible keywords.
- Used LuceneStemmer Library to get the stemmed form of all the words of user's command.
- Built a tree like data structure where every node is a HashMap containing the keywords, and the leaves are all the actions that can be taken.
- Based on the keywords spotted in the user's command, the tree is traversed from root to leaves, and we get the final action to be executed.

  This approach appeared too inflexible, as ordering of the key words might vary, so it's incorrect to fix the nodes at every step while building the tree. Therefore a different data structure was tried.

- A boolean table data structure was built, where rows were all the keywords and columns were all the possible action key sequences.

- For a particular keyword all those entries were made “1”, which occurs in the corresponding action column.
- Then ‘OR’ operation is done between all the rows i.e all the keywords which appears in user’s command, and the column containing all “1” was the required action.

While we were testing the validity and flexibility of this approach, a Microsoft team got interested in our project and advised us to use LUIS which is specifically designed for intent understanding, so we switched from this approach.

Current Approach:

Language Understanding Intelligent Service(LUIS) is a machine learning based Microsoft service which can be used in applications to understand valuable information from text.
LUIS is used to understand the user’s intent from the text obtained after converting speech.

- Data Set is provided in the form of textual commands where the possible intents and entities are provided, and the model is trained.
- The user’s command is sent to LUIS end-point which recognizes it’s intents and entities along with their confidence scores.
- Currently, intent understanding has been done for various elements like paragraph, line, heading, word,list, etc…
- LUIS returns a JSON response.
- This response is then parsed by a Python script, which gives the corresponding action based on the intent.

NVDA Add-On

Initially we tried sending the keystrokes using libraries like SendKeys, pyautogui and expecting the current screen reader to trap those keystrokes, but this didn’t work according to the required behaviour.
When discussed with the NVDA community, they said that “NVDA uses a global keyboard hook rather than responding to Windows messages for keyboard-related events, which may have something to do with your software not working as expected.Relying on keyboard control is fragile because users will often change the keystrokes inside NVDA or use an alternative NVDA keyboard layout (desktop, laptop, etc.).”

So after these unsuccessful attempts and also on the advice of NVDA community, finally we decided to move to the NVDA Add-On approach, where in we could use pre-defined NVDA functions. It required some dependencies to be installed, which are:

- Python version 2.7.15- Current NVDA components are written in Python 2 mostly.
- NVDA screen reader: This can be downloaded from here https://www.nvaccess.org/download/
- Python keyboard module- This is used to send keyboard strokes programmatically and can be installed using the command - pip install keyboard.
- Python speech recognition module-It is a library for performing speech recognition and can be installed using the command - pip install SpeechRecognition.
- Python PyAudio module- This is used to record and play an audio on a variety of platforms and can be installed using the command - pip install pyaudio.

Keyboard Module

Takes full control of your keyboard with this small Python library. Hook global events, register hotkeys, simulate key presses, listens and sends keyboard events.

Current Possible Navigation

The current design supports navigating across next/previous or skipping multiple times at one go, among the following items:

Uses NVDA functions:

- Character
- Word
- Line

Uses Keyboard module:

- Heading
- List
- List Item

- Graphics

It also allows navigating to the following items, which is done using NVDA functions:

- Start of the line
- End of the line
- Top of the page
- Bottom of the page

Not supported

Navigation support for following items is not functional in the current design:

- Paragraph
- Table
- Link
- Also doesn't allow commands like:
  - Go to the third heading (or line or any of the items listed above.)
  - Read the next five lines (or any of the items listed above.)

4. Time Analysis

Approximate time taken at each step when done for average over 5 sentences for each example:

❏ To convert speech to text : 4.764 secs
❏ Intent understanding through LUIS : 1.7 secs

	Add-On Method(sec)	Keyboard(sec)	Manual(sec)
Overall execution time	6.86	6.065	0.15
Only execution time	0.132	0.115	0.15

There was not much difference between navigating across various items.

We see that majority of the time taken is due to online speech to text conversion. It may improve if any offline version with good accuracy can be obtained.
The time taken can further improve if once trained, the model can be made available offline to avoid accessing the end-point of LUIS every time a command is sent. If only the execution time is compared, this module gives the result almost as fast as pressing the keys, which can be seen from the above result.

5. Future Work

- Improve the accuracy of PocketSphinx, or look for other alternatives to convert speech-to-text offline with a good accuracy.
- Support commands like “Go to the second heading”.
- Enabling it to recognise intents for incomplete commands, e.g. “Skip this too”.

6. References

- DictationBridge - https://github.com/dictationbridge
- Simulate Keyboard Inputs - https://docs.microsoft.com/en-us/dotnet/framework/winforms/how-to-simulate-mouse-and-keyboard-events-in-code.
- Microsoft LUIS - https://www.luis.ai/
- NVDA AddOn Development - https://github.com/nvdaaddons/DevGuide/wiki/NVDA-Add-on-Development-Guide