Experimenting with Hotword Detection: The Pao-Pal

Kuo-Pao Yang, Alexander Jee, Dane Leblanc, Jeremiah Weaver, Zachary Armand

Abstract — Hotword detection and voice command are implemented in many of today's current technological devices. In this paper, the implementations using hotword detection as well as the improvements in hotword technology will be discussed including the Pao-Pal. The Pao-Pal was made using Google's AIY Voice Kit which contains a Raspberry Pi Zero WH, voice bonnet, and a speaker that allows it to playback pre-recorded voice lines. These components allow the Pao-Pal to act as a switch for a Keurig machine to make coffee when certain phrases are said. To allow the Pao-Pal to always listen for hotwords, it uses a hotword detection engine called Snowboy, which also allows it to playback the pre-recorded voice lines. Snowboy allows users to create and customize their own special hotword. Other projects concerning voice commands will be discussed along with the application of that kind of technology.

Index Terms — Hotword Detection; Raspberry Pi; Voice Recognition; Google AIY Voice Kit

I. INTRODUCTION

As technology begins to grow and improve over time, computer scientists and engineers are experimenting and trying to integrate voice recognition with their products and ideas. This growth in technology is what spur many new and creative ideas that implement hotword detection and voice commands. Many of the leaders in the tech industry have their own version of a personal assistant, which allows users to interact with their product and complete certain tasks. For example, Apple has Siri, Google has its own Google Assistant, Microsoft has Cortana and Amazon has Alexa. Typically, the way personal assistants work is when a certain hotword or phrase, such as "Ok, Google", is heard the personal assistant recognizes and allows the user to do a variety of tasks. These tasks can vary from giving the user the current time, giving the definition of words, setting alarms, etc.

However, there are certain features some of these personal assistants do not have. With other popular personal assistants, they can be somewhat difficult to implement new voice commands for the user. Also, they do not usually allow you to make a custom voice for the assistant, but with the Pao-Pal the voice can easily be changed and turned into the "G-Doc-Bot."

Although there are many other available technologies that utilizes voice recognition, there are situations where the Pao-Pal is more convenient. The Pao-Pal is built off

Google's AIY project kits and uses Snowboy, a hotword detection engine, which allows the Pao-Pal to be customizable. The Google AIY Voice Kit consists of a Raspberry Pi Zero WH, a speaker, voice bonnet, etc. Since these components are available with the Pao-Pal, it provides an easy way for users to customize and test their ideas. The Pao-Pal's application of Snowboy is what allows the Pao-Pal to have fully customizable hotwords and be completely functional without the use of internet, which is a feature that many other personal assistants do not offer. While using these features, the Pao-Pal can successfully brew a cup of coffee while hooked up to a Keurig machine.

With many other developers creating new ideas for implementing voice recognition, their applications of using similar technology will be discussed. The following papers with their ideas, implementations, and improvements to speech recognition, voice commands, hotword detection, etc. will be summarized with slightly more detail. One project used the same Google AIY Voice Kit without using the voice bonnet that comes the kit; and instead, they used the Raspberry Pi itself and the Google assistant to recognize different languages focusing on Romanian [1]. This also shows how the Google AIY Voice Kit gives users an easy time in branching out and following through with their ideas. There has been also development integrating TV applications for Android OS and Google Assistant [2]. For humans to interact with personal assistants with ease, selecting the correct command for a certain task is very essential [3]. With Google Assistant, they are constantly trying to improve their product. One way they enhance a user's experience with Google assistant is improve the keyword spotting system (KWS) accuracy by using a server-side contextual automatic speech recognition (ASR) [4]. There has also been tests for improving the experience when interacting with other devices that implement voice commands and pointing at certain parts of the devices' screen [5]. With voice command technology, there are ways to improve handicapped quality of life. One research group from Wilbur Wright College experimented with a mobile reading device for people that are blind, and another team created a Sudoku game for blind people as well [6], [7]. Other ideas for improvements to the overall experience of hotword detection and personal assistants involve implementing accelerometer sensors and its data to detect hotwords in order to lower energy costs and increase the assistants hotword detection accuracy [8]. To improve hotword detection in loud or in public area, Google has also tried to use dual microphones in order to filter out the ambient noise and to find just the hotword the user said [9]. There are also studies on endpoint detection, which is an important part of speech recognition that detects the end of each voice command [10].

With our application of Google's AIY Voice Kit, we use

Published on September 29, 2020.
Kuo-Pao Yang
(corresponding e-mail: kyang@selu.edu)
Alexander Jee
Dane Leblanc
Jeremiah Weaver
Zachary Armand

the Raspberry Pi [11] and the special Raspbian operating system to enable special features from the voice kit. In place of the Google Assistant API, the Snowboy hotword detection is used instead. The Raspberry Pi is connected to a Keurig machine which allows the Raspberry Pi to trigger the Keurig machine and brew coffee. Our implementation of hotword detection in our project, the Pao-Pal, will be explained into more detail later, along with the results and testing of our project.

II. RELATED WORK

There are many different approaches to using hotword detection and voice commands, including the improvements to their performance as well. Using the Raspberry Pi that comes with Google's AIY Voice Kit, one research group integrated Google Assistant with the kit to record or capture speech signal and produce an output to generate an audio answer [1]. However, they focused on the speech recognition performance in different languages. The difference between our project and their research is that we did not make use of the Google Assistant API and, instead, used Snowboy for our commands and the voice responses.

With personal assistant technology, interaction between certain applications and devices are improved. For example, there are a couple of different research groups that worked on applications to improve the quality of life of handicapped users. One of them involved trying to create a mobile reading device for visually impaired people. [6]. They came up with certain designs and tests in order to best accommodate to their needs. To allow people with other disabilities to play computer games, a different group made an application that allows users to play Sudoku using speech recognition [7]. Using a Speech Application Programming Interface (SAPI) created by Microsoft, they successfully created an application that allows users to play Sudoku using voice commands.

Combining personal assistant or voice command technology with other devices is a common idea among many developers. One team's idea involved creating an easier and more efficient way of controlling your TV using Google Assistant for the Android Operating System (OS) [2]. Their goal is to improve user's quality of experience, so he/she does not have to use a remote controller but to say a voice command. Another team was testing which is the best method for interaction between humans and machines using voice commands [5]. The result of those tests was that users preferred the shorter, concise, and more human-machine like interaction compare to the human-human.

There are also many studies and tests involving the speed, efficiency, and accuracy of speech recognition. Determining when the user stops talking and ends the command is a very important part of speech recognition. There has been research involving the tests used to create the best method to accurately determine when the user's command ends. One example is a team creating an algorithm that more accurately determines the word endpoints or simply the end of a sentence [10]. They tested their method by obtaining data from real world environments. When a device has multiple commands that can accomplish similar or different tasks, there can be confusion for users to decide which

command to use, which is why the Information Systems Department from the University of Maryland, Baltimore County (UMBC) has researched this topic [3]. There has also been tests of improving hotword detection using an accelerometer sensor [8]. Using the sensor and its data that is found on many current mobile devices, the accuracy of detecting hotwords have increased while also lowering energy costs.

Google has also been finding new ways to improve their own personal assistant and its performance. Google has been testing out their own hotword cleaner which implements dual-microphone noise reduction [9]. What this does is exploits two unique properties of hotwords. These assumed properties of hotwords involve the leading phrase of valid voice query and that the hotword has a short duration. The final result is ignoring loud TV background noise. Another method Google experimented with is using a keyword spotting system (KWS) that uses contextual automatic speech recognition (ASR) [4]. Using the contextual ASR increases the accuracy of the KWS which allows the user to speak seamlessly, without pausing between the trigger phrase and the voice command.

III. IMPLEMENTATION

A. Software Approach and Implementation

Our project involves using the Raspbian operating system on the Raspberry Pi that came with the Google AIY Voice Kit. It also includes AIY software that allows us to make use of the LED button. For our Raspberry Pi to detect hotwords through the microphone, the Pao-Pal uses Snowboy. Snowboy is also used to play back the pre-recorded voice lines once certain hotwords are detected. Since we are using a Raspberry Pi, our code is made in Python. The following code in Fig. 1 shows our implementation of Snowboy for a function that is called when certain hotwords are detected.

```
def clockCycle():
    leds.pattern = Pattern.breathe(500)
    leds.update(Leds.rgb_pattern(Color.GREEN))

    snowboydecoder.play_audio_file("resources/final/clock_cycle.wav")
    leds.update(Leds.rgb_on(Color.CYAN))
    enddetector.terminate()
    main()
```

Fig. 1. Function called after "Hey Pao-Pal" and "What Time is it?" is said.

The clockCycle function, once called, selects the color and the pattern that the LED button blinks. Then, it plays the pre-recorded audio by using Snowboy decoder function. Once that happens, the resting color of the LED button is changed back to cyan.

In order for our program to use the Snowboy decoder and the AIY software, it first imports those specific packages. The following code in Fig. 2 lets our program implement the board, the LED button, and most importantly, the Snowboy decoder.

To connect to our project, we used the AIY Project application, which gave us the IP address of our AIY Voice Kit. This gave us control of our project via laptop.

Creating the hotwords for our project was very simple.

Snowboy has a website that allows us to easily create a simple .pmdl file. Once the file is on the Raspberry Pi, Snowboy uses the .pmdl file as data to detect when the user says the hotword.

```
import snowboydecoder
import sys
import signal
import time
from aiy.board import Board, Led
from aiy.leds import Leds, Color, Pattern
```

Fig. 2. Code that allows the Pao-Pal to use Snowboy and the AIY.

B. Hardware Approach, Equipment, and Programming

The hardware that is used involves the Raspberry Pi shown in Fig. 3. It has a voice bonnet and LED button from the AIY Voice Kit, a relay, and a Keurig machine. The goal of this project is to use the microphone from the Raspberry Pi to trigger certain commands and turn on and brew coffee using a Keurig machine. In Fig. 4, certain pins on the Raspberry Pi are used in order to connect it to the AIY Voice Kit materials and the relay which triggers the Keurig machine.

Fig. 3. Raspberry Pi Zero WH.

PIN NAME	FUNCTION	Voice Bonnet	FUNCTION	PIN NAME
	3V3 power	1	2	5V power
GPIO 02	I2C SDA	3	4	5V power
GPIO 03	I2C SCL	5	6	Ground
GPIO 04	GPIO	7	8	UART TXD
	Ground	9	10	UART RXD
GPIO 17	GPIO	11	12	I2S BCLK
GPIO 27	GPIO	13	14	Ground
GPIO 22	GPIO	15	16	GPIO (LED button)
	3V3 power	17	18	GPIO
GPIO 10	SPI MOSI	19	20	Ground
GPIO 09	SPI MISO	21	22	GPIO
GPIO 11	SPI SCLK	23	24	SPI CE0
	Ground	25	26	SPI CE1
GPIO 00	I2C (EEPROM)	27	28	I2C (EEPROM)
GPIO 05	GPIO	29	30	Ground
GPIO 06	GPIO	31	32	PWM0
GPIO 13	PWM1	33	34	Ground
GPIO 19	I2S LRCLK	35	36	GPIO
GPIO 26	GPIO	37	38	I2S SDIN
	Ground	39	40	I2S SDOUT

Fig. 4. Pins on Raspberry Pi Zero WH.

For our project, the Raspberry Pi Zero WH that comes with the Google AIY Voice Kit is used. The following code in Fig. 5 initializes the pins from the Raspberry Pi and the LED and assigns them to certain variables in our program. This allows us to control the LED button and the pins that connect to the relay that triggers the button on the Keurig.

All of these components allow the Pao-Pal to light up the LED and control the Keurig using custom hotwords.

```
from aiy.pins import PIN_A, PIN_B
from gpiozero import LED
coffeegpio = LED(PIN_A)
drinkgpio = LED(PIN_B)
leds = Leds()
```

Fig. 5. Code assigning pins on Raspberry Pi.

After hooking up all Keurig to the Pao-Pal, we needed to create a method that powers on the pin, which triggers the relay and starts brewing the coffee shown in Fig. 6. The function used to power the button on the Keurig is very similar to the 'clockCycle()' function that plays the recording. However, the difference is that the 'coffee' function activates the pin connected to the relay which acts as a button for the Keurig. For us to connect the Keurig to the Pao-Pal, it required some soldering work.

```
def coffee():
    leds.pattern = Pattern.breathe(500)
    leds.update(Leds.rgb_pattern(Color.YELLOW))
    snowboydecoder.play_audio_file("resources/final/here.wav")
    leds.update(Leds.rgb_on(Color.CYAN))
    coffeegpio.on()
    time.sleep(1)
    coffeegpio.off()
    endedetector.terminate()
    main()
```

Fig. 6. Code turning on pin connected to relay.

For each hotword, a function will be called similar to Fig. 1 and Fig. 5. Each function selects the pattern in which the LED blinks and selects the color. Once that is selected, the audio file specific to that hotword is played. At the very end of the function, the main function is called. This allows the Pao-Pal to stay on for the duration of the demo. The only function that turns off the program is the function for the hotword "goodbye" which instead of calling the main function, it simply exits the program. The following code in Fig. 7 shows the function called when "Hey, Pao-Pal Goodbye" is said. The Pao-Pal is connected to the Keurig shown in Fig. 8.

```
def end_run():
    leds.pattern = Pattern.breathe(500)
    leds.update(Leds.rgb_pattern(Color.RED))
    snowboydecoder.play_audio_file(
        "resources/final/happy_ending_byebye.wav")
    leds.update(Leds.rgb_off())
    exit()
```

Fig. 7. Function called to end demo.

IV. EVALUATION

Although there are many available personal assistants, there are not many that allow the ease of customizability the Pao-Pal offers. The voice responses can easily be changed with a few lines of code and using the audio files of your choice. When running our project, the detection of the hotwords were slightly inconsistent which gives other personal assistants an advantage. There are also more features available to users when they use Google Assistant

or Siri, which is something the Pao-Pal lacks. However, changing the hotwords for certain commands is a feature that many other personal assistants do not have. There are many features that can be added to the Pao-Pal but to match the number of features other modern personal assistants have will take much more time.

Fig. 8. The Pao-Pal connected to the Keurig.

The possibilities that our project offers are endless and is easy to create a few simple commands and tasks. Our project does allow an easy way of implementing new, simple ideas. It also allows simple customizations that may be difficult with other personal assistants. For example, the triggering hotword for the Google Assistant can only be “Ok, Google” or “Hey, Google”. To implement a new hotword with our project, all we would need is to create a .pmdl file using Snowboy’s website. Once that .pmdl file is on our Raspberry Pi, it is very simple to implement that into the project.

Overall our project does not exceed in performance compared to other personal assistants, as the detector can be somewhat inconsistent due to the microphone on the Raspberry Pi being subpar. However, it does allow for ease of customization and implementing certain ideas.

The Pao-Pal also successfully joined with another group to create a voice command drink machine using Arduino [12], [13]. It has a few motors and tubing that sucks up a reservoir of drinks and combines them. The Pao-Pal is connected to the drink machine similar to how we connected to the Keurig. We used a relay to power up the button to power the motors for the drink machine. Therefore, the function that will be used is almost exactly the same as the “coffee()” function shown in Fig. 9. The one difference is that a different pin is used.

```
def drink():
    leds.pattern = Pattern.breathe(500)
    leds.update(Leds.rgb_pattern(Color.YELLOW))
    snowboydecoder.play_audio_file("resources/final/here.wav")
    leds.update(Leds.rgb_on(Color.CYAN))
    drinkgpio.on()
    time.sleep(1)
    drinkgpio.off()
    enddetector.terminate()
    main()
```

Fig. 9. Function called to power drink machine.

V. CONCLUSION

As technology continues to grow rapidly, personal assistant technology becomes more common in the people’s everyday lives. The application of personal assistant, hotword detection, and voice command can be used in many different ways. However, customizing more mainstream personal assistants such as Google Assistant, Siri, and Alexa can be difficult. Using Snowboy, the Pao-Pal allows the user to customize voice commands. Through Snowboy, the Pao-Pal can also create custom hotwords, which can easily be done by creating a .pmdl file using Snowboy’s website. What the Pao-Pal currently does is allow the user to play back specific voice recordings and power a Keurig machine. The Keurig is powered by the Raspberry Pi that comes with Google’s AIY Voice Kit. To power the Keurig, the Raspberry Pi is connected to a relay that powers the button and brews the coffee. By using Google’s AIY software, the Pao-Pal can customize the pattern in which the LED blinks and change its color.

Albeit not perfect, the Pao-Pal definitely accomplishes the tasks it was created to do. It successfully uses Snowboy and the AIY software to allow us to customize and implement hotword detection. It also allows us to playback the audio recordings that were made for responses to certain hotwords.

REFERENCES

- [1] S. Mischie, L. Mățiu-Iovan and G. Gășpăresc, "Implementation of Google Assistant on Raspberry Pi," *2018 International Symposium on Electronics and Telecommunications (ISETC)*, Timisoara, Romania, pp. 1-4, 2018.
- [2] A. Lazić, M. Z. Bjelica, D. Nad and B. M. Todorović, "Google Assistant Integration in TV Application for Android OS," *2018 26th Telecommunications Forum (TELFOR)*, Belgrade, Serbia, pp. 420-425, 2018.
- [3] J. Feng, "Modeling command selection for speech-based applications," *In CHI '03 Extended Abstracts on Human Factors in Computing Systems (CHI EA '03)*, pp. 680-681, 2003.
- [4] A. H. Michaely, X. Zhang, G. Simko, C. Parada, and P. Aleksic, "Keyword spotting for Google assistant using contextual speech recognition," *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, Okinawa, Japan, pp. 272-278, 2017.
- [5] M. Elepfandt, "Pointing and speech: comparison of various voice commands," *In Proceedings of the 7th Nordic Conference on Human-Computer Interaction: Making Sense Through Design (NordCHI '12)*, pp. 807-808, 2012.
- [6] R. Keefer, S. Narayanan, and N. Bourbakis, "Voice commands for a mobile reading device for the visually impaired," *In Proceedings of the 3rd International Conference on Pervasive Technologies Related to Assistive Environments (PETRA '10)*, no. 3, pp. 1-4, 2010.
- [7] L. Zhang, P. Pathak, M. Wu, Y. Zhao, and P. Mohapatra, "AccelWord: Energy Efficient Hotword Detection through Accelerometer," *In Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '15)*, pp. 301-315, 2015.
- [8] Y. A. Huang, T. Z. Shabestary, and A. Gruenstein, "Hotword Cleaner: Dual-microphone Adaptive Noise Cancellation with Deferred Filter Coefficients for Robust Keyword Spotting," *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '19)*, Brighton, United Kingdom, pp. 6346-6350, 2019.
- [9] P. K. Pal and S. Phadikar, "Modified energy based method for word endpoints detection of continuous speech signal in real world environment," *2015 IEEE International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, Kolkata, India, pp. 381-385, 2015.
- [10] S. Norte and F. Lobo, "A sudoku game for people with motor impairments," *In Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE '08)*, pp. 319-319, 2008.

- [11] K. P. Yang, T. Beaubouef, and M. Chiu, "Lesson Learnt from Smart Home Automation Systems," *Journal of Emerging Trends in Computing and Information Sciences*, ISSN 2079-8407, vol. 6, no. 3, pp. 149-153, March 2015.
- [12] K. P. Yang, G. Kiepper, B. Henry, and R. Hunter, "Modular Architecture for IoT Home Automation and Security Surveillance," *Journal of Multidisciplinary Engineering Science and Technology (JMEST)*, ISSN 2458-9403, vol. 5, no. 11, pp. 8978-8982, November 2018.
- [13] K. P. Yang, P. McDowell, P. Dolan, C. Otts, G. Chenevert, and C. Tunstall, "SelfieBot: A Robot to Detect, Photograph, and Tweet Smiles," *International Journal of Engineering Research & Technology (IJERT)*, ISSN 2278-0181, vol. 8, no. 10, pp. 387-390, October 2019.

Kuo-pao Yang is a full professor in the Computer Science department at Southeastern Louisiana University. He received his B.S. degree in Computer Science at Tamkang University, Taipei, Taiwan, R.O.C. in 1991. He earned his M.S. degree in Computer Science from Illinois Institute of Technology in 1994. He received his Ph.D. degree in Computer Science at Illinois Institute of Technology in 2003. His research interests include Computer Architecture, Programming Languages, and Expert Systems.