

Washington University in St. Louis

Washington University Open Scholarship

All Computer Science and Engineering
Research

Computer Science and Engineering

Report Number:

2019-5

Smart Home Audio Assistant

Xipeng Wang

This report introduces an audio processing algorithm. It provides a way to access smart devices using audio. Although there are many audio assistants already on the market, most of them will not be able to control the smart devices. Therefore, this new system presented in this report will provide a way to analysis the customer's questions. Then the algorithm will be able to query smart device information, modify the schedule or provide the reason for some arrangement.

Follow this and additional works at: https://openscholarship.wustl.edu/cse_research



Part of the [Computer Engineering Commons](#), and the [Computer Sciences Commons](#)

Recommended Citation

Wang, Xipeng, "Smart Home Audio Assistant" Report Number: (2019). *All Computer Science and Engineering Research*.

https://openscholarship.wustl.edu/cse_research/1173

Department of Computer Science & Engineering - Washington University in St. Louis
Campus Box 1045 - St. Louis, MO - 63130 - ph: (314) 935-6160.

Smart Home Audio Assistant

Xipeng Wang

Abstract

This report introduces an audio processing algorithm. It provides a way to access smart devices using audio. Although there are many audio assistants already on the market, most of them will not be able to control the smart devices. Therefore, this new system presented in this report will provide a way to analysis the customers questions. Then the algorithm will be able to query smart device information, modify the schedule or provide the reason for some arrangement.

1 Introduction

With the development of AI technology, more and more smart devices appear in real life. Such as cleaning robot, smart refrigerator and smart speaker. The increment of intelligent products raises one problem that it is difficult to schedule and control them. Although Amazon Echo and Google Home can solve some part of this question, it will be better to have a system or assistant can help customers manage these machines.[FYP17]

Therefore, a smart home audio assistant is implemented to figure out the problem. First, let us assume that there is a scheduler that can gather information from multiple smart home devices. This schedule producer can generate a flexible schedule with some constraints to minimize the electricity cost.[FYP17] The purpose of this project is that let customers be able to modify and query the schedule using voice control. This voice controller algorithm must be robustness to the audio input and provide regular feedback based on the requests from users.

2 Algorithm

There are four main parts of the algorithm. The first part is the voice input part, which will use the microphone to collect users audio input and transform it into text. Then the text information will be passed to the second part: text analysis. Text analysis is a key part of this project. It will decompose the texts and analyze the words to figure out the customers intent using NLP algorithm. This part will produce a standard output to make the next step easier. The part after the text analysis is querying the schedule. This querying part will collect corresponding information from the schedule and generate a human-understandable text reply using the standard information from the last step. The last step is translating the text to audio and play it to users.

```

73 func Recording(name string) {
74     i := MCIWorker("open new type waveaudio alias capture", "", 0, 0)
75     if i != 0 {
76         log.Fatal("Error Code A: ", i)
77     }
78 }
79
80 i = MCIWorker("set capture alignment 2 bitspersample 16 samplespersec 44100 channels 1 bytespersec 88200", "", 0, 0)
81 if i != 0 {
82     log.Fatal("Error Code S: ", i)
83 }
84
85 i = MCIWorker("record capture", "", 0, 0)
86 if i != 0 {
87     log.Fatal("Error Code B: ", i)
88 }
89
90 fmt.Println("Listening...")
91
92 time.Sleep(6 * time.Second)
93
94 i = MCIWorker("save capture " + name + ".wav", "", 0, 0)
95 if i != 0 {
96     log.Fatal("Error Code C: ", i)
97 }
98
99 i = MCIWorker("close capture", "", 0, 0)
100 if i != 0 {
101     log.Fatal("Error Code D: ", i)
102 }
103
104 }

```

Figure 1: The code for recording the audio

```

206 func Recognizing(name string) string {
207     ctx := context.Background()
208     // Creates a client.
209     client, err := speech.NewClient(ctx)
210     if err != nil {
211         log.Fatalf("Failed to create client: %v", err)
212     }
213     // Sets the name of the audio file to transcribe.
214     filename := "C:/Users/xwa24/cse_project/" + name + ".wav"
215
216     // Reads the audio file into memory.
217     data, err := ioutil.ReadFile(filename)
218     if err != nil {
219         log.Fatalf("Failed to read file: %v", err)
220     }
221     // Detects speech in the audio file.
222     resp, err := client.Recognize(ctx, &speechpb.RecognizeRequest{
223         Config: &speechpb.RecognitionConfig{
224             Encoding:    speechpb.RecognitionConfig_LINEAR16,
225             SampleRateHertz: 44100,
226             LanguageCode: "en-US",
227         },
228         Audio: &speechpb.RecognitionAudio{
229             AudioSource: &speechpb.RecognitionAudio_Content{Content: data},
230         },
231     })
232     if err != nil {
233         log.Fatalf("failed to recognize: %v", err)
234     }
235     // Prints the results.
236     for _, result := range resp.Results {
237         for _, alt := range result.Alternatives {
238             fmt.Printf("%v" (confidence=%3f)\n", alt.Transcript, alt.Confidence)
239             temp.WriteString(alt.Transcript)
240         }
241     }
242     return(temp.String())
243 }

```

Figure 2: The code for translating audio to text

2.1 Voice input to text

There are two easy steps for this part. Firstly, the algorithm will use the `mciSendString` function provided by the Windows system to access the systems microphone. This microphone will record the voice input and store it to a single channel .wav file because the next step will only take single channel audio file as input. In Figure 1, line 80, the variable named channels is 1. Then in Figure 1 line 92, the audio capture length is set in seconds. The code showing in Figure 1 will generate a .wav audio file collected from the microphone with a given name.

The second step for this part is transforming the audio input to text output. The Google Cloud provides an API called Speech-to-Text. This API enables developers to convert audio to text. Figure 2 shows the code using this API to convert the file. In line 214, the variable means the path to the audio file to transcribe. From line 222 to line 231 will set the features of the audio file as we collected in the previous step. At last, the code will get the response from Google Cloud API and return the text as function output.

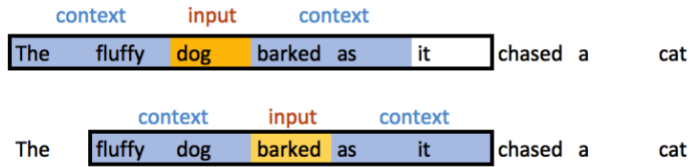


Figure 3: A example of Word2Vec

```

octopus.path_similarity(octopus) # 1.0
octopus.path_similarity(nautilus) # 0.33
octopus.path_similarity(shrimp) # 0.11
octopus.path_similarity(pearl) # 0.07

```

Figure 4: Some examples of WordNet Similarity results

2.2 Text analysis and NLP algorithm

Now, the text translation from the users audio input is available. Text analysis aims to find out customers aiming. So, three keywords must be recognized: device name, action, and question type. As the algorithm should be robust enough to handle different questions, word similarity will be helpful. One of the most well-known algorithms for word similarity is Word2Vec. Word2Vec will generate a word matrix where each row is a vector of that word. For every word in the given text, Word2Vec will scan through the whole text with a fixed length window.[Ron14] As shown in Figure 3, the blue words are inside the moving window, and the target word is yellow. The algorithm will count the frequency that words appeared inside the window. After scanning a large corpus of text, each unique word will have its vector. The similarity of the two words is the distance between two vectors. However, there are a few drawbacks of Word2Vec. The input must be a large corpus of text to generate an accurate matrix. The other disadvantage is that one word will only have one vector which means if this single word has multiple meanings, the word vector will not be able to deal with it.[LM14]

For this project, one word may have multiple meanings. For example, the word water can indicate both a water machine and water tank. The other problem is that there is not enough text to train this particular word matrix. Therefore, to avoid these two issues, WordNet is the solution. A WordNet is a dictionary. It stores each word by their meanings which followed by all the words synonyms.[PPM04] Same as the regular dictionary, different kinds of words, such as verb adj, are separated. The similarity between the two words is calculating by the words listed below each target word. Figure 4 shows some examples.

```

device.txt - Notepad
File Edit Format View Help
dyson, heater, heat, cold, hot
bryant, cooler, cool, hot, cold
rheem, tank, water
roomba, robot, cleaner, vacuum
tesla, car, vehicle, drive, Tesla
washer, wash, laundry, water
dryer, dry, laundry, water
baker, bake, broiler, broil, oven, kenmore, Baker
dishwasher, washer, kenmore, dish, dishes, water

```

(a) A sample dictionary for devices

	heater	dryer	water	washer
heater	1	0	0	0
dryer	0	1	1/4	0
water	0	1/4	1	2/4
washer	0	0	2/4	1

(b) A sample similarity matrix

Figure 5: Example for similarity algorithm

Questions	[Device Action Question Time]
What is the schedule of my car	[tesla none sche none]
Why my heater on at 5	[dyson on why 5]
Turn my oven off at 14	[baker off set 14]

Figure 6: Some examples of text processing

The new algorithm is combining these two well-known methods. As mentioned before, to make a standard output, three components must be recognized. Therefore, the same as the WordNet, three dictionaries are built. The device dictionary is shown in Figure 5(a). Each line is the same meaning words pointing to a single device. As presented in Figure 5(a), there are nine different devices, and each device has multiple similar words. With this dictionary, a matrix can be computed using the following algorithm. For every unique word in this dictionary, there will be a value compared to all words. If two words are the same, the value will be 1. If not, the value will be:

$$value(a, b) = \frac{\# \text{ of } a \text{ and } b \text{ appeared in the line}}{\# \text{ of } a \text{ or } b \text{ appeared in the line}}$$

where a and b denotes two words. A sample matrix is built and shown in the Figure 5(b). For example, the $2/4$ value in the row washer, whose denominator is the number of lines that washer or water is showing in the dictionary. That is line three, six, seven and nine. The nominator is the two words showing in the same line, which is line six and nine. Therefore, the value is $2/4$. Similarly, three dictionaries and matrices are computed.

With the similarity algorithm, the input text is processed and decomposed into four keywords output. Three examples in Figure 6. The second column is the output for this part, and it will be used to query the schedule.

Section	Action	Value
0:	ts:	1
	bl:	0.2
	ep:	0.198
	off:	1
	wash:	0
	dec:	0
	dbl:	-1
1:	ts:	2
	bl:	0.16
	ep:	0.198
	off:	1
	wash:	0
	dec:	0
	dbl:	-1

(a) A sample JSON schedule

```

402 // Loop though all possible
403 v := reflect.ValueOf(myschedule).FieldByName(name)
404 fmt.Println(time)
405 for i := 0; i < v.Len(); i++ {
406     sche := v.Index(i)
407     if i == time-1 {
408
409         a0 := sche.FieldByName(actions[0])
410         a1 := sche.FieldByName(actions[1])
411         fmt.Println(a0,a1)
412         if a0.Int() == 0 && actionin == "off"{
413             a0.Set(reflect.ValueOf(1))
414             a1.Set(reflect.ValueOf(0))
415             break
416         } else if a0.Int() == 1 && actionin == "on"{
417             a0.Set(reflect.ValueOf(0))
418             a1.Set(reflect.ValueOf(1))
419             break
420         } else {
421             changable = false
422             break
423         }
424     }
425 }

```

(b) The code query from the schedule

Figure 7: Example for querying the schedule

2.3 Query schedule

The schedule is in JSON form, and a sample schedule is in Figure 7(a). The query code is in Figure 7(b). The name variable in line 403 is coming from the device keyword. The same kind of variables is in line 407 and 409, where comes from action and time. The question keywords are used to decide which kind of questions that user is asking, and not in this part of this code.

2.4 Voice output

Enough information should be collected in the last step and combined into text output using the values from the schedule. This text can be uploaded to Google Translate using an HTTP request. The website will translate the text into an mp3 audio file. Same as the audio input part, the mciSendString function is used to play the audio file using the systems speaker.

3 Test and results

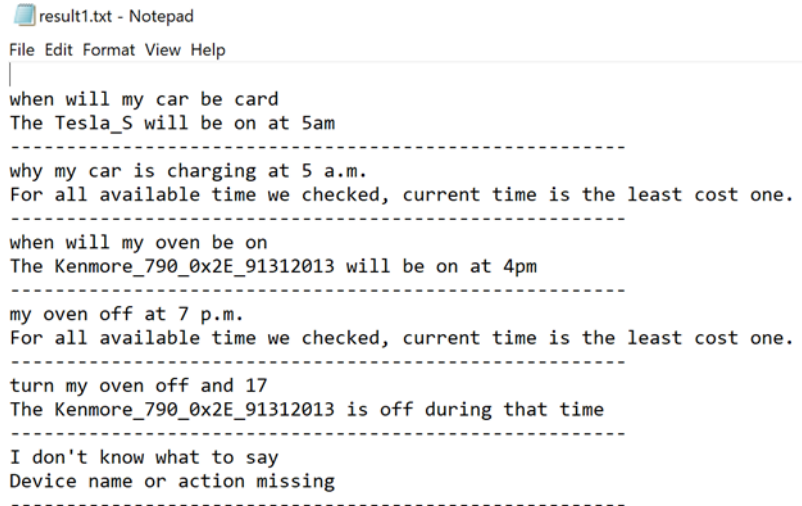
3.1 Robustness test

As described in the introduction part, this algorithm should be robust enough to deal with different formats of questions. So, the test questionnaire will not specify the format of asking questions. Therefore, the test will provide a scenario description, which is a small story. Then ask the testers to ask the system for some information. By setting up questions in this way, the questionnaire will be able to test the robustness of this algorithm. A sample question is in Figure 8

You need to drive your car to work tomorrow. And, so, you want to know the charging schedule of your Tesla car? Please ask the system to provide that information.

Ps: Answer should provide you the schedule of your car

Figure 8: A sample question in the questionnaire



```
result1.txt - Notepad
File Edit Format View Help
|
when will my car be card
The Tesla_S will be on at 5am
-----
why my car is charging at 5 a.m.
For all available time we checked, current time is the least cost one.
-----
when will my oven be on
The Kenmore_790_0x2E_91312013 will be on at 4pm
-----
my oven off at 7 p.m.
For all available time we checked, current time is the least cost one.
-----
turn my oven off and 17
The Kenmore_790_0x2E_91312013 is off during that time
-----
I don't know what to say
Device name or action missing
-----
```

Figure 9: A response of the questionnaire

3.2 Results

The program will store the users input and the system output as a log file. In Figure 9, the first question the system not recognize the users audio correctly, but the response is right. The system made a mistake in the fourth question. Because the system does not record the keyword "turn", the response is wrong. The correct answer is in the fifth question. At last, when a user asks something unrelated, the system response is in the sixth question.

4 Future development

For the whole system presented in this report, it is not smart enough to deal with all situations. For example, this similarity method will not be able to understand not on command. Apart from that, the similarity matrix algorithm combining two well-known methods is only my simple idea. There must be better ways to optimize it. Moreover, the neural network is also an excellent way to do the users intent analysis. At last, the response can be smarter. When some essential part is missing, the system may ask the user some question to provide the missing value to full fill the query.

5 Appendix

This is link to the code: <https://github.com/MinuteMaidJuice/Audio-Assistant>

References

- [FYP17] Ferdinando Fioretto, William Yeoh, and Enrico Pontelli. A multiagent system approach to scheduling devices in smart homes. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pages 981–989. International Foundation for Autonomous Agents and Multiagent Systems, 2017.
- [LM14] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196, 2014.
- [PPM04] Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. Wordnet:: Similarity: measuring the relatedness of concepts. In *Demonstration papers at HLT-NAACL 2004*, pages 38–41. Association for Computational Linguistics, 2004.
- [Ron14] Xin Rong. word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*, 2014.