Development of Voice Commands in Digital Signage for Improved Indoor Navigation Using Google Assistant SDK

David Sheppard, Nick Felker, and John Schmalzel, Ph.D.
Dept. of Electrical and Computer Engineering
Rowan University
Glassboro, NJ, USA
sheppardd1@students.rowan.edu, felker@google.com, schmalzel@rowan.edu

***Abstract*—One of the most challenging aspects of trying to navigate in a new indoor place can be making sense of the navigational tools available. Traditionally, static maps have been used to provide guidance to people in indoor spaces. Recently, a new category of signage has begun to gain prominence in this area: digital signage. With the advent of such devices, many approaches have been taken to provide information to people in a more useful way. Some digital signs simply provide information in a predetermined manner, while others allow for user interaction using buttons or touchscreens. Although many digital signs allow for a more dynamic display of information, few are able to easily provide custom information based on easily acquired user input. We propose a novel type of digital signage that is voice-forward, using voice input and natural language processing to allow the users to find information faster than a standard touchscreen-only modality.**

***Index Terms*—Android, Google Assistant, indoor navigation, IoT, voice ui, conversational design**

I. Introduction and Objectives

While indoor navigation has posed a challenge for most people in an unfamiliar environment, recent technology has shown a promise in improving indoor way-finding. Traditionally, static signs have been placed throughout public places to help those who are new to the environment understand their whereabouts. While those signs can be useful, they do not provide an easy method for updating their information or providing custom information tailored to those using the sign. An attempt is being made to make use of digital signage to help provide a larger set of on-demand information.

A. Digital Signage

Digital signage is a relatively recent development in the world of navigation. Digital signs are television-like devices that can be used to display advertisements, maps, menus, and much more. They can be found in malls, fast food restaurants, and even on gas pumps [1]. Some of them are interactive, incorporating touch screens or implementing physical buttons for users to interface with. According to [1] published in 2012, "LCD displays have decreased in cost more than tenfold over the past decade," making deployment of digital signs much more economical than in the past. As of 2012, it was also estimated that the digital signage market generated about $5 billion [1]. With a market this big, many companies have arisen with attempts to solve common problems with digital signs.

One such digital sign in development is the Smart Sign. Unlike most current digital signs, the Smart Sign is able to vocally interact with the user, allowing the sign to address a user's specific needs. The Smart Sign is a small device running the Android operating system that uses the power of the Google Assistant to handle spoken queries. The device has a touchscreen and uses a small microphone to accept verbal input.

In addition to supporting all of the Google Assistant's built-in queries, the Smart Sign extends the Google Assistant to serve as a navigation aide like traditional signage. The user can ask for directions to a room and the Smart Sign will provide them with a map of the building with a route plotted on the map leading from their current location to their requested destination. Additionally, the device can easily be modified to accept new commands as the need for them arises. This allows the device to be continually improved upon even after being deployed in a building.

B. Related Work

While many have attempted to improve indoor navigation by way of technology, most have assumed that the answer lies in mobile phones. Typically, this method involves estimating the user's current location within the building and providing directions based on their current location. While mobile phone location methods like GPS can be very precise outdoors, they often fall short of providing accurate information in buildings [2], leaving developers to attempt other positioning methods.

Some have attempted to use the accelerometer and compass found in many mobile phones to provide navigation assistance to users. Approaches like these often make use of dead reckoning, a method that involves frequently estimating location based on inputs like step detection [3]. While this may

The Android Things device used for development and testing of the Smart Sign was provided for free by Google.

978-1-5386-5541-2/18/$31.00 ©2018 IEEE

help to overcome some of the deficiencies in current indoor positioning methods, it still requires the user to have a phone that is capable of accurately determining their location and orientation. Likewise, methods like this can quickly accumulate errors. These errors must then be compensated for using yet another method [2], [3].

Attempts have been made to counteract or eliminate these errors using other surrounding signals. In [4], RSS values published at individual shops within a mall were used to provide accurate location data. Others have attempted to use geomagnetic signals to determine a user's location [5]. Still others have attempted to employ the user's turns to provide correction to dead reckoning algorithms as the person is in motion [2].

Recently, the Android operating system has implemented a solution that involves the use of Wi-Fi networks to help a phone determine its location. Implemented in Android 9.0, this method uses the Wi-Fi Round-Trip-Time (RTT) application program interface (API) to estimate a compatible phone's "distance to nearby RTT-capable Wi-Fi access points" [6]. This provides a software-based solution for the user, but it still requires them to be within range of compatible hardware. Regardless of error mitigation methods, it can be challenging to give directions within a building in a way that is both accurate and user-friendly.

C. Voice Interaction

The ability to quickly perform voice transcription and natural language processing has enabled the creation of voice assistants like Google Assistant [7], Alexa [8], and others. Some of these voice assistants provide software development kits (SDKs) for hardware manufacturers and hobbyists to embed this technology in their own Internet of Things (IoT) devices.

By introducing voice transcription, the user of a device is able to find what they're looking for with a single voice command instead of having to either scan a static screen or navigate an unfamiliar user interface.

We propose a novel type of digital signage that is voice-forward, using voice input and natural language processing to allow the users to find information faster than a standard touchscreen-only modality.

II. Design Methodology

Unlike many other indoor navigation methods which rely on a user's mobile phone to guide them, the Smart Sign is a device that exists in a fixed location. Since the device is stationary, its location can be known with a high degree of precision. This eliminates the problematic need to accurately track the user as they navigate. As a result, the user no longer needs to rely on their phone and its sensors in order to properly navigate a building. Instead, a single device is able to provide succinct directions to the user in the likeness of a traditional map.

A. Android Things

The device used for development is the Android Things Pico Pro Maker kit [9]. This device features a Pico i.MX7D development board along with a 5-inch touch display, a Wi-Fi antenna, and a USB-C power cable. The board features an "implementation of two ARM® Cortex®-A7 cores" operating at up to 1.2 GHz along with an "ARM® Cortex®-M4 core" [9]. The board features pins that use voltages ranging from 1.8 V to 5 V [9]. The device also requires use of a power source that can output at least 1 A in order to function.

The Android Things platform is designed as an IoT device that is simple to develop for. The use of the Android operating system along with the ability to add additional input and output devices allows for more efficient development of IoT devices and prototypes [10]. Apps can be easily updated on the device and new features can be taken advantage of when released in regular Android software updates.

As one of the most popular operating systems in the world [11], Android development skill is not uncommon and many people are familiar with the operating system. During testing, the device was running Android 8.1, one of Android's most current operating systems. The familiarity of the Android environment to both programmers and users helps to make development and use of the Smart Sign much less daunting. Using the popular Android framework allows for programmers to build upon existing libraries already in place [10], thus streamlining the development process.

The Smart Sign employs the Android Things platform to produce a device that is not unlike a mobile phone running the Android operating system. For portability, a case that is able to contain the components in a compact form was 3D printed using design files that were found online [12]. A small USB microphone was connected to the device's USB port and external speakers were connected to the device's standard 3.5 mm audio port. Fig. 1 shows the device in its case with the USB microphone attached.

Fig. 1. The Smart Sign in the idle state

B. Google Assistant SDK

Another advantage to using the Android platform is the availability of many of Google's API libraries and develop-

ment platforms for use in the application. This includes the Google Assistant, Google's virtual assistant that can respond to users' text and verbal queries using natural language processing. The Google Assistant SDK for devices [7] provides an API that allows software to programmatically make calls to the Google Assistant on behalf of a user.

This is accomplished with a gRPC service [13]. gRPC is a framework for streaming data between a client and server. The message format is based around protocol buffers [14], a data serialization format. A separate compiler application, protoc, [15] can be used to generate language-specific implementations which can be used in this application. The protocol buffer files that define the Google Assistant SDK service are publicly available. In this Android project, a build command is employed to generate the interface code for sending protocol buffers through gRPC.

The Google Assistant has many built-in queries and actions and provides ways for additional actions to be added by developers for their own use [7]. The Smart Sign uses custom-defined actions in an action package [16] that can aid a user in indoor navigation. Some of the custom commands that were defined include phrases like "Where is room ___?" which allows the user to ask for a specific room number within the building. Using the Google Assistant allows for Google's commands to be built upon, rather than starting from scratch.

Custom commands are defined using an action package [17]. An action package is a JavaScript Object Notation (JSON) file that stores custom commands which can be referenced by the main code when the Google Assistant is called. Action packages allow the developer to specify desired variables within the spoken phrases that may change with each request. For example, the phrase "Where is room ___?" may be used frequently with different room names. By using a variable for room numbers, the code can act on user's specific request and produce an output that is specific the value of the variable. Additionally, the phrases can be written in such a way that synonyms (e.g. room and classroom) are added as optional words in a phrase. With this method implemented, requests like "Where is room ___?" can be considered the same as "Where is classroom ___?" in the code.

The essential contents of the action package can be seen in Appendix A. The package contains potential queries and defines variables within the queries that may produce different responses based on their values. (For the sake of brevity, only a few of the potential phrases for finding rooms have been included.) While the phrase definitions are given in the actions.json file, the logic that determines the application's response to the queries is located in the application's primary source code (not shown).

C. User Interface

The program itself is an Android application. It is currently compatible with Android 8.1 and above. The graphical user interface (GUI) of the application is designed to be simple and intuitive to the user. The application starts out in an idle state which can be used to display information such as the weather, advertisements, and alerts when the device is not actively taking requests. While in the idle state, the user can initiate a request by tapping an icon on the screen to open the assistant activity.

The voice user interface (VUI) is present in the assistant activity. When this activity launches, the device begins recording audio input through the attached microphone. When it appears that the user has finished speaking, the device stops recording and provides a response to the user's query. If the user asks for directions within the building, the GUI responds by displaying a map of the building with a polyline leading from the user's current location to their desired destination. An example of this can be seen in Fig. 2. If the user asks a question unrelated to navigation, the Google Assistant will respond verbally by way of the speakers. After a 40 second period of inactivity, the application returns to the idle state.

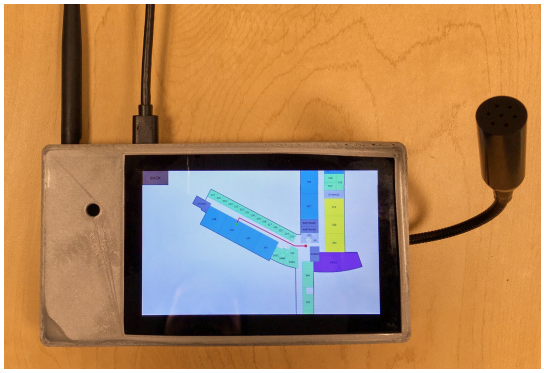

Fig. 2. Custom directions given in response to a user's query

III. Testing and Evaluation

In order to test the Smart Sign, the device was deployed on the third floor of a building on the campus of Rowan University. Students were encouraged to test the sign and complete a survey on their experience. The sign was available for alpha testing for about one week. During that time, over 200 queries were processed. The categories of requests can be seen in Fig. 3.

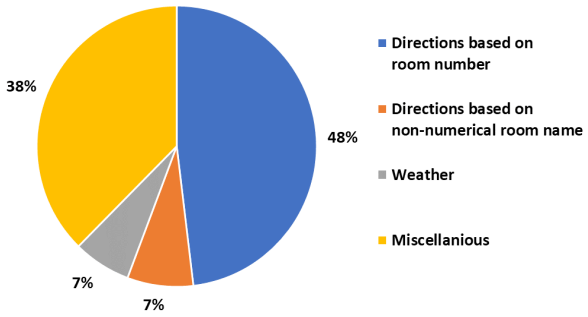

Fig. 3. Requests made to Smart Sign during alpha testing

For the purpose of testing, the Smart Sign was displayed on a music stand with computer speakers placed on the stand on

either side of the sign. The use of a music stand allowed for people of various heights to adjust the stand to better suit them if they desired. On the music stand was a paper explaining the purpose and intended use of the sign. During the testing period, the device was available for testing 24 hours a day.

After the testing period was over, the survey results were analyzed and the user's queries were viewed. The queries were viewed using the "My Activity" webpage [18] of the Google account that the Smart Sign was signed into during the testing period. This allowed for every query interpreted by the Google Assistant to be viewed along with the Google Assistant's response to each request.

IV. Results and Discussion

After the alpha testing phase was complete, the survey and query results were analyzed and potential improvements were theorized. One such problem that users encountered was the fact that the microphone that was used during testing required the user to be relatively close to it; they typically had to be within one foot of the microphone for the Google assistant to understand them. With further analysis, it the microphone's range showed to be the bottleneck of the VUI. When tested with a laptop computer, it became clear that the microphone could not pick up sound that came from much further than one or two feet away. Therefore, it was concluded that better signal processing would not significantly improve voice recognition. Instead, a new microphone with a higher sensitivity must be used to improve user experience.

Another problem that was encountered was the software's misinterpretation of the users' queries. Sometimes, the software had trouble recognizing the word "room," substituting it for words like "route". Additionally, the need to add rooms with non-numerical names (e.g. "office" and "atrium") was further underscored by the users' queries. Finally, it was discovered that some additional custom commands needed to be added to accommodate for unexpected phraseology that was sometimes used by testers when asking for directions to a room.

Despite the Smart Sign's downfalls, many users were pleased with the results. The ability to receive quick and succinct directions to a room was appealing, while the integration of the Google Assistant allowed them to ask questions that they are accustomed to using with other digital assistants. With further development, many of the current problems will likely be overcome.

V. Conclusions

Overall, the benefits of a Smart Sign became apparent during development and testing of the device. The need for quick and succinct directions is often realized by many people when trying to navigate an unfamiliar place. The testing and feedback demonstrated that the Smart Sign shows promise in aiding indoor navigation of the future. While there are many more commands and features that must be added to the application, the Smart Sign is still under development to ensure that it can be easily scaled for use in larger applications.

The code software is open source, and can be found at https://github.com/rowanpwlab/smart-signs.

VI. Further Work

Three primary goals have been set for the future of the Smart Sign: to expand the database of custom commands, to implement server-side integration of the code, and to add features to the idle state of the application. Expanding the database of custom commands will make the Smart Sign more user-friendly by allowing the Assistant to be more effective at responding to needs of the users. Server-side integration of necessary code will allow for the application to be updated in a much easier manner. By storing items like indoor maps on a server, developers will be able to easily update directions and maps as needed without having to completely reinstall the app on the device.

Lastly, the idle state should be revamped to include relevant information such as announcements, current events in the building, advertisements, and even suggestions on how to use the Smart Sign. While the device's current idle state contains information about the weather, implementing a series of items to display could make the device more helpful to users. In addition to these improvements, further testing of the Smart Sign with a new microphone will help to ensure relevancy and promote further improvements.

References

[1] R. Want and B. N. Schilit, "Interactive Digital Signage," in *Computer*, vol. 45, no. 5, pp. 21-24, May 2012.

[2] A. Masiero, A. Guarnieri, F. Pirotti, and A. Vettore, A Particle Filter for Smartphone-Based Indoor Pedestrian Navigation, *Micromachines*, vol. 5, no. 4, pp. 1012–1033, Nov. 2014.

[3] J. A. B. Link, P. Smith, N. Viol and K. Wehrle, "FootPath: Accurate map-based indoor navigation using smartphones," in 2011 International Conference on Indoor Positioning and Indoor Navigation, Sept. 2011, pp. 1–8.

[4] K. Dong, W. Wu, H. Ye, M. Yang, Z. Ling, and W. Yu, Canoe: An Autonomous Infrastructure-Free Indoor Navigation System, *Sensors*, vol. 17, no. 5, p. 996, Apr. 2017.

[5] D. Liu, S. Guo, Y. Yang, Y. Shi and M. Chen, "Geomagnetism Based Indoor Navigation by Offloading Strategy in NB-IoT," in *IEEE Internet of Things Journal*.

[6] "Wi-Fi location: ranging with RTT," *developer.android.com*, Oct. 4, 2018. [Online]. Available: https://developer.android.com/guide/topics/connectivity/wifi-rtt. [Accessed Nov. 19, 2018].

[7] "Google Assistant SDK for Devices," *developers.google.com*, Aug. 8, 2018. [Online]. Available: https://developers.google.com/assistant/sdk/overview. [Accessed Nov. 11, 2018].

[8] "Alexa Voice Service Device SDK," *developer.amazon.com*, Nov. 21, 2018. [Online]. Available: https://developer.amazon.com/alexa-voice-service/sdk. [Accessed Nov. 21, 2018].

[9] "NXP i.MX7D," *developer.android.com*, Sept. 25, 2018. [Online]. Available: https://developer.android.com/things/hardware/imx7d. [Accessed Nov. 18, 2018].

[10] "Android Things," [Online]. Available: https://developer.android.com/things/. [Accessed Nov. 19, 2018].

[11] "Operating System Market Share Worldwide: October 2017 - October 2018," *statcounter.com*, Oct. 2018. [Online]. Available: http://gs.statcounter.com/os-market-share. [Accessed Nov. 15, 2018].

[12] OttoES, "NXP PICO MX7D with touch screen and camera case," *thingiverse.com*, Oct. 11, 2017. [Online]. Available: https://www.thingiverse.com/thing:2580821. [Accessed Sept. 6, 2018].

[13] "What is gRPC," *grpc.io*, [Online]. Available: https://grpc.io/docs/guides/. [Accessed Nov. 21, 2018].

[14] "Protocol Buffers — Developer Guide," *developers.google.com*, [Online]. Available: https://developers.google.com/protocol-buffers/docs/overview. [Accessed Nov. 21, 2018].
[15] "Protocol Buffer Basics: Java," *developers.google.com*, [Online]. Available: https://developers.google.com/protocol-buffers/docs/javatutorial. [Accessed Nov. 21, 2018].
[16] "Device Actions Overview," *developers.google.com*, Oct. 2, 2018. [Online]. Available: https://developers.google.com/assistant/sdk/device-actions-overview. [Accessed Nov. 14, 2018].
[17] "Register Custom Device Actions," *developers.google.com*, Oct. 9, 2018. [Online]. Available: https://developers.google.com/assistant/sdk/guides/library/python/extend/custom-actions. [Accessed Nov. 20, 2018].
[18] M. Williams, "How to check your Google Assistant history," *techhive.com*, para. 3-4, Apr. 24, 2018. [Online]. Available: https://www.techhive.com/article/3268921/digital-assistants/how-to-check-delete-google-assistant-history.html. [Accessed Nov. 18, 2018].

APPENDIX

A. Action Package

Below is the actions.json file that was used in the project to process user queries and extract key entities.

```
{
  "manifest": {
    "displayName": "Room Number",
    "invocationName": "Room Number",
    "category": "PRODUCTIVITY"
  },
  "actions": [
    {
      "name": "com.acme.actions.Room_Number",
      "availability": {
        "deviceClasses": [
          {
            "assistantSdkDevice": {}
      }]},
      "intent": {
        "name": "com.acme.intents.Room_Number",
        "parameters": [
          {
            "name": "number",
            "type": "SchemaOrg_Number"
          }
        ],
        "trigger": {
          "queryPatterns": [
          "Where is (classroom)? (room)? $SchemaOrg_Number:number (at)?",
          "Where can I find (classroom)? (room)? $SchemaOrg_Number:number (at)?",
          "How would (you)? (I)? get to (classroom)? (room)? $SchemaOrg_Number:number"
      ]}},
      "fulfillment": {
        "staticFulfillment": {
          "templatedResponse": {
            "items": [
              {
                "deviceExecution": {
                  "command": "com.acme.commands.Room_Number",
                  "params": {
                    "number": "$number"
    }}}]}}}},
    {
      "name": "com.acme.actions.Room_Name",
      "availability": {
        "deviceClasses": [
          {
            "assistantSdkDevice": {}
      }]},
      "intent": {
        "name": "com.acme.intents.Room_Name",
        "parameters": [
          {
            "name": "name",
            "type": "RoomName"
          }
        ],
        "trigger": {
          "queryPatterns": [
            "Where is (room)? $RoomName:name",
            "How do I get to (room)? $RoomName:name"
      ]}},
      "fulfillment": {
        "staticFulfillment": {
          "templatedResponse": {
            "items": [
              {
                "deviceExecution": {
                  "command": "com.acme.commands.Room_Name",
                  "params": {
                    "name": "$name"
  }}}]}}}}],
```