JustSpeak: Enabling Universal Voice Control on Android

Yu Zhong[1], T.V. Raman[2], Casey Burkhardt[2], Fadi Biadsy[2] and Jeffrey P. Bigham[1,3]

Computer Science, ROCHCI[1]
University of Rochester
Rochester, NY, 14627
zyu@cs.rochester.edu

Google Research[2]
Mountain View, CA, 94043
{raman, caseyburkhardt,
biadsy}@google.com

Human-Computer Interaction Institute[3]
Carnegie Mellon University
Pittsburgh, PA, 15213
jbigham@cmu.edu

ABSTRACT

In this paper we introduce *JustSpeak*, a universal voice control solution for non-visual access to the Android operating system. *JustSpeak* offers two contributions as compared to existing systems. First, it enables system wide voice control on Android that can accommodate any application. *JustSpeak* constructs the set of available voice commands based on application context; these commands are directly synthesized from on-screen labels and accessibility metadata, and require no further intervention from the application developer. Second, it provides more efficient and natural interaction with support of multiple voice commands in the same utterance. We present the system design of *JustSpeak* and describe its utility in various use cases. We then discuss the system level supports required by a service like *JustSpeak* on other platforms. By eliminating the target locating and pointing tasks, *JustSpeak* can significantly improve experience of graphic interface interaction for blind and motion-impaired users.

ACM Classification Keywords

H.5.2 Information Interfaces and Presentation: User Interfaces—*Voice I/O*; K.4.2 Computers and Society: Social Issues—*Assistive technologies for persons with disabilities*

General Terms

Human Factors, Design

Author Keywords

Universal voice control, accessibility, Android, mobile

INTRODUCTION

Mouse and multi-touch surface have been widely used as the main input methods on computers and mobile devices for their reliable accuracy. But under circumstances when visual access to the display is impossible or hindered, or for users with dexterity issues, it is difficult to point at a target so they are often less effective. For blind and motion-impaired people this issue is more obvious, but other people also often face this problem, e.g, when driving or using a smartphone under bright sunshine. Voice control is an effective and efficient alternative non-visual interaction mode which does not require target locating and pointing. Reliable and natural voice commands can significantly reduce costs of time and effort and enable direct manipulation of graphic user interface.

In graphical user interfaces (**GUI**), objects, such as buttons, menus and documents, are presented for users to manipulate in ways that are similar to the way they are manipulated in the real work space, only that they are displayed on the screen as icons. The first and most essential step of interaction with GUI is target locating, sighted people can intuitively find the target object with a quick visual scan. But without visual access to the screen, this simple task is often very hard to finish, which is the main cause of difficulties in non-visual interaction. Before the invention of screen readers [1, 7, 6], it was essentially impossible to interact with a GUI non-visually. Now that audio feedback has been used to support non-visual accessibility of computer and smartphones, many works [2, 4] have been focusing on utilizing voice control to improve the efficiency of eyes-free user input. As for motion-impaired people, they face challenges in the second step, point targeting, as most GUI interaction techniques like mouse and multi-touch require accurate physical movements of users' hands or fingers.

The advantage of voice commands over mouse or multi-touch when interacting with a screen non-visually is that it does not require targets to be located and thus avoids the problems with pointing as described before. This task often costs the majority of time needed to complete interaction with an input device, for example, moving a cursor onto a button with a mouse. This has been deeply studied since at least the introduction of Fitts's law [11] in the 1960s. On the contrary, when using speech as input, users can expect a computing device to automatically execute commands without the hassles of finding and pointing on-screen objects. For example, to complete a simple task like switching off Bluetooth on a multi-touch smartphone, with speech-enabled interface,

W4A '14, April 7-9, 2014, Seoul, Korea

a user can simply tell the phone to *"switch off Bluetooth"* instead of going through the time consuming process of finding and clicking the settings icon, and then the Bluetooth switch.

Unfortunately, existing voice control systems are constrained in many ways. We have not yet seen a system which supports universal voice commands on a device able to control any application. The majority of them are either built on application level (e.g. speech input methods) or limited by pre-defined commands set. For instance, even the most popular voice assistants like Siri and Google Now [2, 4] can only support built-in functionalities on the device like messaging, calling and web searching, etc.

In this paper, we present *JustSpeak*, which is designed and built with a fundamentally different architecture on the system level of Android. *JustSpeak* aims at enabling universal voice control on Android operating system to help users quickly and naturally control the system non-visually and hands-freely. Users can easily access *JustSpeak* on their existing Android smartphones and tablets. The application runs in the background as a system service which can be activated with simple gestures, a button or a Near Field Communication (**NFC**) tag no matter which application is running in the foreground. When activated, *JustSpeak* records the speech spoken by the user. The recorded audio is then transcribed into plain texts and parsed into computer understandable commands which are automatically executed by *JustSpeak*. Unlike most current systems, *JustSpeak* can accomodate commands for any application installed on the device, for example, clicking buttons, scrolling lists and toggling switches. Moreover, *JustSpeak* is smarter and more natural than other works in that it supports multiple commands in one utterance. For instance, to open the Gmail application then refresh the inbox, two commands can be combined into once sentence *"Open Gmail then refresh"*. The two main features differ *JustSpeak* from other voice control systems and offer more intuitive and efficient eyes-free and hands-free interaction.

The contributions of this paper include:

- a mobile application, *JustSpeak*, that enables universal voice control on Android devices;
- a description of the system design that empowered accommodation of adaptive commands parsing and chaining; and
- a discussion on leveraging speech input to enhance interaction for blind people and people with dexterity issues.

RELATED WORK

Non-visual Interaction

As described before, currently nearly all computing devices adapt GUI to offer friendly and natural interaction. However, GUI was essentially quiet and inaccessible for non-visual users before the introduction of screen readers [12]. Over the last two decades, many screen readers have been successful in providing accessibility for blind users on multiple platforms. For example, JAWS [7] on Windows personal computers, WebAnywhere [3] for the web, VoiceOver [1] on Macintosh computers and iOS, TalkBack [6] on Android devices, etc. Those softwares identify and interpret what is being displayed on the screen and then re-present the interface to users with text-to-speech, sound icons or a braille output device. With the non-visual feedback of screen readers, blind people can explore the user interface, locate and manipulate on-screen objects.

Screen readers largely improved the accessibility of GUI for blind users. However, blind users still need to adapt to traditional input techniques that require target discovery. Those techniques are designed for visual interaction, hence require additional efforts for blind people. For example, on non-touch devices, blind people often use keyboard shortcuts to iterate through the interface linearly (from top to bottom, left to right) in order to find an object, and use gestures similarly on touch interfaces.

For sighted people who are not familiar with screen readers, there are few works that allow them to interact with their devices non-visually. Mostly they have to wait until visual access to the devices is possible. Although the non-visual interaction problem is often merely nuisance for them, sometimes it can cause great inconvenience, for instance, when someone wants to check the map while driving a vehicle.

Speech Recognition Services

Speech recognition, also known as speech-to-text or automatic speech recognition (**ASR**), has been studied for more than two decades [14] and recently been used in various commercial products ranging from speech input like customer service hotlines, voice IMEs, to automatic virtual agents such as Siri and Google Now which will be discussed in the following section.

Recognition of natural spoken speech is a very complex problem because vocalizations vary interms of accent, pronunciation, pitch, volume, speed, etc. The performance of a speech recognition engine is usually evaluated in terms of accuracy and speed. There are a large number of ASR services, both in academia as research projects [9, 13], and in industry as commercial products [8, 10].

In the framework of *JustSpeak*, we employ Google ASR service [10] to recognize spoken speech of users. Google has been working on speech processing for a long time and their voice recognizer is widely used in multiple products across different platforms, including voice search on mobile phones and webpages, voice IMEs and Youtube [5] video closed captioning. Their application programming interface (**API**) enables easy and seamless integration of scalable text-to-speech functionality

in mobile applications, and supports both online and offline speech recognition.

The performance of Google ASR service is also top notch within the competitive area. When using online recognition, the word error rate (**WER**) is around 15 percent, and the average real time (**RT**) factor is below 1 second [10]. It empowers *JustSpeak* to provide users fast and accurate voice interaction experience.

Voice Control Systems

We define voice control systems as applications or devices controlled by means of human voice. Unlike direct utilization of voice recognition such as voice IMEs or voice search, voice control systems apply the results of voice recognition to manipulate user interface or execute specified commands. All mainstream smart mobile phones now support different levels of voice control, including Android, Microsoft Windows phones, Blackberry and iOS. [16] The most advanced and popular voice control applications are Google Now on Android and Siri on iOS.

Google Now is a dialogue system on Android (4.1 and above) that supports execution of a set of actions on the device, including functions of most Google products. For example, users can set reminders by saying *"Remind me to call John at 6PM"*, and similarly create calendar events, show locations on Google Maps, call or message a contact, ect. [4] with Google Now.

Apple firstly introduced simple voice control into iOS as a new feature of iOS 3, they then partnered with Nuance [8] to create bring Siri into iPhone 4S and newer models as a replacement. Users of Siri can issue voice commands similar to but less than Google Now. For example, Siri does not support functions of the map application on iOS.

Although Google Now and Siri are both considered the state of art voice control systems, they are both limited by the way they define their supported commands. Essentially each command can only have one key word or phrase which specifies a certain function supported by the system, for instance, *"calendar"* or *"alarm"*. If there is no key word found in the sentence or the sentence does not satisfy the grammar associated with the key word, the whole sentence will be treated as a web search query. Also, if a user wants to execute two or more voice commands, s/he has to repeat the process several time because they can only process one command in each dialog turn. Furthermore, since the functions supported are pre-defined and built into the applications, they do not work with applications outside the constrained scope, for example, third party email software.

Android Accessibility APIs

Android accessibility APIs [15] give developers ability to provide better accessibility to users who have special needs. With the APIs, Android developers can make their own application more accessible by offering audio prompts, physical feedback and alternative navigation modes. They can also make their own accessibility services which run in the background and provide those enhancements for all applications, a set of application or just a single app. For example, TalkBack, the default screen reader of Android, is an accessibility service.

Given permissions from a user who installed an accessibility service, developers can access the on screen view hierarchy, accessibility events and enhance user experiences accordingly through the accessibility APIs. Those APIs are powerful tools because they act as a delegate between applications and system so that an accessibility service can be aware of any interaction performed by the user and interface changes triggered by the event. For example, when an user launches an Android app, accessibility APIs will notify registered accessibility services and attach all the view objects shown in the application activity including buttons, labels, menus, etc. to the notification. Applications can also leverage accessibility APIs to fire interaction events on behalf of the user, including but not limited to, clicking, long pressing and scrolling an actionable on-screen object.

JUSTSPEAK

JustSpeak is an Android accessibility service designed for use with any existing applications and other accessibility services (e.g. TalkBack) on smart phones and tablets running Android 4.2 and up. The interaction to launch applications and activate on screen control via spoken commands is simple and intuitive and fully accessible for non-visual use. It has been released on Google Play Store for free downloads, for now *JustSpeak* only supports English.

In this section, we will first discuss how *JustSpeak* handles speech that contains only one command and describe the system designs which enable multiple commands in one sentence in the last part.

System Description

Same as other Android accessibility services (e.g. TalkBack), once installed, *JustSpeak* can be enabled under *Settings -> Accessibility* (shown in Figure 1), this enabling operation only needs to be performed once as *JustSpeak* will be automatically restarted when the device is rebooted. The same accessibility setting page is also the place to turn off *JustSpeak*.

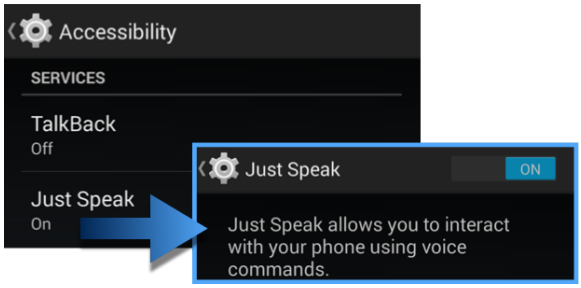

Figure 1. Enabling *JustSpeak* in Accessibility Settings

As shown in Figure 2, *JustSpeak* has three modules working together to facilitate universal voice commands once activated by the user. The first step is speech recognition which transcribes spoken speech into plain text. Then the utterance parsing module kicks in to process the voice recognition results into formalized commands. And finally we search for actionable objects on the current screen with the command arguments and manipulate the highest matching object or initiate system commands accordingly.

JustSpeak is efficient in terms of battery consumption because it runs in background sleeping mode while not being used. When the user wants to issue voice commands, *JustSpeak* can be easily activated in several different ways, including dragging the home button up, clicking the *JustSpeak* application icon and scanning a customized NFC tag. Then an earcon is played to alert the user *JustSpeak* is ready to record audio, a green layer on the right edge of the screen will also be displayed to show volume changes of detected audio with color variation. A challenge we faced when designing *JustSpeak* is that many on-screen objects are iconized and do not have labels shown beside them. For example, the four application icons on the bottom of screen as shown on the first phone screen in figure 2. In order to let users easily discover what can be said, *JustSpeak* will briefly illustrate the label associated with each actionable object on top of it after activated.

Speech Recognition

As discussed before, we used the Google ASR service in *JustSpeak* to recognize user input speech. In addition to the reliable performance, Google ASR gives developers great flexibility by offering both offline and online recognition, therefore, *JustSpeak* can be used without internet connection. Of course, there are advantages of connecting to the Google servers. The most important benefit for *JustSpeak* users is that online recognition returns multiple scored results as opposed to single result when using offline recognition. A simple example is shown in table 1, when the user said "setting", Google ASR returned three different results. This feature give *JustSpeak* ability to increase speech recognition error tolerance and voice commands success rate because we can iterate through all the results by descending scoring order and execute the most likely action. Details about the process will be discussed in the following section.

Table 1. Multiple results with scores returned from Google online ASR when saying *"setting"*

Recognition Result	Score
setting	0.90
settings	0.08
sitting	0.02

Utterance Parsing

JustSpeak enables voice control of all the interface manipulating functions supported by the Android accessibility APIs as well as some global system commands that can be invoked programmatically. A complete list of currently supported actions and examples are shown in table 2 below.

Table 2. Voice actions supported by *JustSpeak*

Action	Example	Synonym
Local commands		
Activate controls	Click reset	Tap, touch
Scroll e.g. lists	Scroll up	forward
Toggle switches	Switch on data	toggle
Long press controls	Hold refresh	long tap
Toggle checkboxes	check vibration	uncheck
Global commands		
Launch apps	Open Hangouts	Launch, run
Show recent apps	Recent Apps	Recents
Show quick settings	Quick settings	Open
Toggle WiFi	Switch WiFi On	Toggle
Toggle Bluetooth	Bluetooth On	Toggle
Toggle tethering	Tethering On	Toggle
Home	Go home	home screen
Back	Go back	
Show notifications	Notifications	
Easy labels	Easy labels	

In order to provide natural and flexible user experience, we designed a grammar based utterance parsing process for *JustSpeak*. Each action can be expressed in several different ways just like communicating with real humans. For instance, to long press a button labeled as "reset", users can tell *JustSpeak* to *"long click/press reset"*, *"press/click and hold reset"* or simply *"hold reset"*. Some synonyms of each supported action are also shown in table 2. In addition, *JustSpeak* also supports insertable words and phrases such as *please, hmm, can you*, etc. We also defined some hot words that triggers commonly used applications, including *"browser/web"* to open default web browser, *"OK Google Now"* to launch Google Now and *"voice search"* or *"search"* to initiate voice search.

When *JustSpeak* receives the scored results from ASR, it tries to process each result with the defined grammar. Once a voice recognition result passes our grammar check, we wrap it up in the form of command name and arguments, then pass it to the execution module along with the recognition score. In the case all of the recognition results of a user speech do not satisfy any grammar listed in table 2, *JustSpeak* will abort and notify the user about the failure with text-to-speech feedback.

Commands Execution

Once commands are parsed, *JustSpeak* tries to execute each of them one by one. Most of the global commands can be executed straightforwardly by simply invoking specified system events. On the other hand, local commands have arguments associated with on-screen controls and accessibility metadata. Therefore, we need to employ accessibility APIs to locate the interactive object the user wishes to access when speaking local commands.

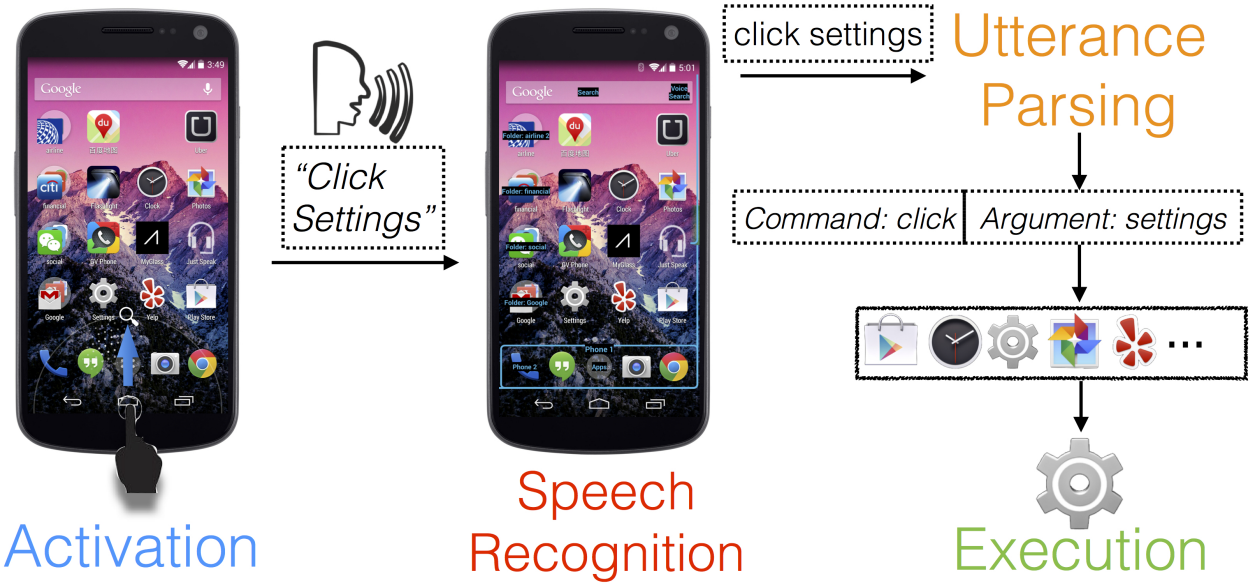

Figure 2. When activated, *JustSpeak* records audio spoken by the user, transcribes it into plain text, then parses the text into formal commands, andn then finally finds the correct object on the screen to perform the command.

To do so, we maintain a view indexer in *JustSpeak*, this indexer is essentially a hashmap mapping the text labels or content descriptions to their associated interactive interface element. For example, if an image button is labeled as *'compose'* by the developer, even if this label is not shown on the screen, we will keep an entry of word *'compose'* mapping to this button. Since the contents shown on the screen are constantly updated. *JustSpeak* listens to all types of accessibility events [1]. Those events include the source of updated view and metadata such as time stamp, details of the modification, identification of corresponded application, etc. They empower *JustSpeak* to dynamically update the indexer to keep it fresh. For example, when the text inside a textbox is modified by the user or the system, *JustSpeak* will receive a *View Changed* event, then *JustSpeak* will swap the stale node and its descendants in the indexer with the newer ones passed in with the event.

This indexer becomes handy when a local command is handed to the execution module, we can query the indexer with the argument string to find matching nodes shown on the screen. Since user inputs are spontaneous and do not always match the labels defined by developers, it is necessary to design a flexible mechanism that allows flexibility. We used a word overlapping based ranking algorithm so that users do not have to say the exact label to find an on-screen object. For example, if a command specifies *'compose message'* and there are two buttons on the screen that contains word *'message'*, then *'compose a new message'* will yield higher score than *'show message'* and *JustSpeak* will perform the command on the *'compose a new message'* button.

[1] Android accessibility events, *http://developer.android.com/reference/android/view/accessibility/AccessibilityEvent.html.*

This algorithm is used for all local commands that require the user to specify name of an interface element, including activation, switch toggling, long pressing and checkbox updating as shown in table 2. Once a unique node is found, *JustSpeak* will validate whether the command can be performed on this specific object with accessibility APIs, if not, it will continue to a lower ranked object. For instance, checking command can be only applied on checkboxes. For scrolling commands, *JustSpeak* only needs to find a scrollable item in the indexer because in most mobile interfaces the chance of more than one extended views existing together is minimum.

As described before, when using online ASR, there are usually several scored commands, *JustSpeak* tries to process each of them to find out whether their arguments correspond to a unique actionable node in the indexer by descending scoring order. This way if ASR produced errors in the highest scored result because of users' accents or other factors, *JustSpeak* still has a large chance of executing the requested command. If none of the command has argument that can be validated in the indexer, this execution attempt will be considered failed. The result of execution is announced with synthesized speech.

Chaining of Commands

Support of multiple commands in single speech is an important feature of *JustSpeak* for two reasons. Firstly, it is more time efficient to combine multiple commands into one sentence than repeating the whole dialog turn for several times; secondly, it is more natural and consistent with the way spontaneous speeches are produced to express ordered requests in single utterance. An example of using chained voice commands is illustrated in figure 3.

In the *JustSpeak* framework, utterance parsing and commands execution modules work together to understand speeches containing more than one commands. All of the supported functions listed in table 2 can be chained into a single sentence. As shown in figure 3, the utterance is parsed into an array of commands in the order that they are placed in the sentence. A challenge placed upon utterance parsing is disambiguation, for example, sentence *'click reset and home'* can be viewed as either one command: clicking button *'reset and home'* or two commands: clicking button *'reset'* and then navigating back to home screen. In order to obtain better disambiguation performance, grammars of each supported action have to be defined as detailed as possible, *JustSpeak* also assigns higher priority to commands that can be validated on current screen, taking the sentence before as an example, if there is a button labeled as *'reset'* on the screen, then the two commands result will be preferred. Since execution of an action usually causes numbers of interface updates, for example, clicking a button often results in opening of a new page, a dialog or an application, the array of commands can not be executed at the same time. In fact, the execution module only tries to validate the first command arguments and execute it if possible, and then waits for a short time until all the accessibility events fired by the previous command are settled and then proceeds to the next command on the updated screen. Theoretically, users can chain as many commands as they wish, but in practice, we found speech recognition errors build up quickly with the growth of speech length. With the error tolerance mechanism discussed before, *JustSpeak* is able to reliably support normal length spontaneous speeches containing four or less commands.

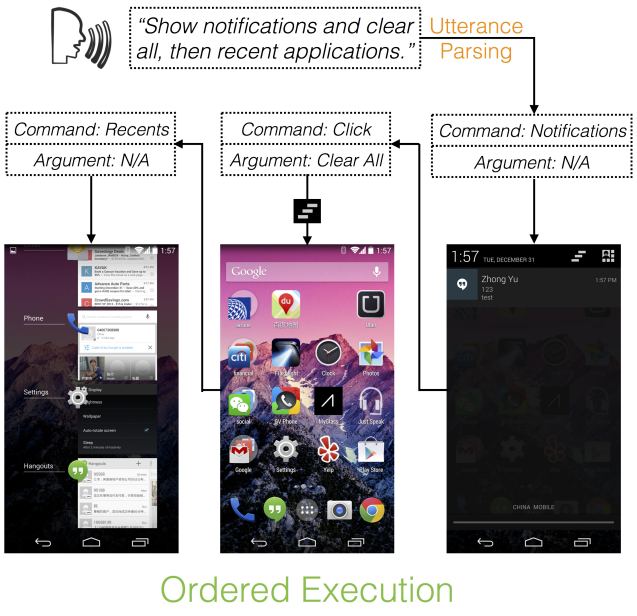

Figure 3. *JustSpeak* parses a spoken sentence into three ordered commands and executes them one at a time.

USE CASES AND DISCUSSION

JustSpeak innovatively provides enhancements to all Android applications and the Android system itself by the means of natural and fast voice control that can be accessed non-visually and hands-freely across the whole platform. Its value is not limited to assisting blind users, in fact all Android device users can benefit from *JustSpeak* in variety of cases.

For Blind Users

As the primary user group of non-visual interaction techniques, we believe blind users of Android devices can interact with their smart phones faster and more easily with assistance of *JustSpeak*. In fact, this project was initially designed specifically for blind users, we only discovered its value for other user groups later during the course of developments.

As described before, most blind people use screen readers to interact with their computers and portable electronic devices. On Android smart phones and tablets, *JustSpeak* does not interfere with existing screen readers and other accessibility services (e.g. TalkBack). On contrary, they work together to offer a better user experience to blind users. In fact, since blind users perceive the representation of user interfaces in the form of text read by screen readers, they are already aware of the strings associated with each object that can be said as valid voice commands. Therefore, they can more easily familiarize themselves with *JustSpeak* and get the best out of it. We have observed that in many cases blind users spend a large amount of time looking for a specific object on their phones. The main reason is that screen readers are designed as a linear iterative screen explorer, although blind users can usually identify an on-screen control by its first few words and fast forward with increased text-to-speech speed, it still takes more time and efforts for them to locate a target on the screen than for sighted users even if they know it's shown on the screen. In unfamiliar applications this problem can be even worse. *JustSpeak* can significantly reduce the time and efforts needed to access application controls for blind users. A simple case that blind users of Android can benefit greatly from *JustSpeak* is launching applications. Nowadays one can easily find hundreds of applications installed on a personal Android device, even with visual access to the screen, it is usually a nightmare for users to fumble through pages of application icons and spot one of them that they want to open. With the assistance of *JustSpeak*, this operation can be as easy as saying the application name. In fact, after the release of *JustSpeak*, we have noticed many users, both blind and sighted, use it to launch applications regularly.

For Sighted Users

Although GUI is designed for sighted users primarily, sighted users often find them inevitably placed under situations where non-visual interaction is required. Without visual access to the screen, and being not screen

reader users, their smartphones or tablets are very difficult to interact with in those cases. Examples like eyes-busy status (e.g. driving, meeting) and environments which render the screen invisible or untouchable (e.g. direct sunshine, wearing gloves) have been described before. *JustSpeak* is an effective alternative to complete urgent tasks without looking at the screen under those circumstances.

From the user feedback we collected, we discovered another unexpected user group which also benefited from *JustSpeak*. One of our users pointed out that *JustSpeak* is also a great tool for *"anyone with dexterity issues"*. Although those Android users do not have difficulty finding on-screen objects, it is often a hard task for them to point to them accurately with fingers on multi-touch screens, *JustSpeak* alleviates their burden to move the device so that they can just place it at a place where they can activate *JustSpeak*, and use spoken commands to let *JustSpeak* perform interactions on their behalf. *JustSpeak* is especially useful for motion-impaired users when completing a complex task because with chained voice commands they only have to finish one physical interaction with the device which is activation. To better server the users with dexterity issues, *JustSpeak* added a 'Tap and Speak' option in the most recent version. Once selected, users can activate *JustSpeak* by touching any place on the screen.

For Android Power Users

Mobile devices are limited by the screen size comparing to computers, therefore mobile app developers adapt hierarchical interface to arrange application contents in different pages. For Android users who are familiar with their applications and know the controls on each application page that need to be accessed in order to complete a complex task , *JustSpeak* can save them hassles to click through several levels to find commonly used functions by chaining commands together. An example is starting a video call with a contact inside Google Hangouts which is a three step task as shown in figure 4 below. With touch interaction, the user has to first launch Google Hangouts at the bottom left corner on the main screen, then click on the contact in the middle, and finally tap the small video call button on the top right corner, with *JustSpeak*, the task is simplified to an activation gesture and one single utterance.

In addition, we also allow users to enable automatic reactivation in *JustSpeak* preferences. Once enabled, *JustSpeak* will be automatically activated to take user input after the previous stack of voice commands are all executed. So that users are able to seamless navigate different applications with voice commands hands-freely. To save battery power, *JustSpeak* is turned off after 5 seconds if it does not detect any voice.

Discussion

As discussed above, we expect that all Android users would benefit from the convenient voice control offered

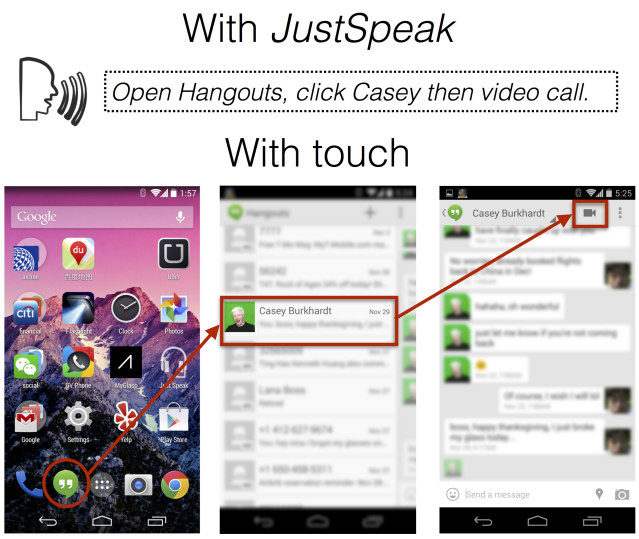

Figure 4. *JustSpeak* can save interaction time to complete a sequence of touch tasks for Android power users.

by *JustSpeak*, therefore we have released it on Google Play Store for free as a beta test version[2]. We did not perform formal user studies or experiments because like other voice recognition based works, the performance of *JustSpeak* largely depends on the performance of ASR service which is affected by many factors. Experiments are difficult to control and essentially evaluating the ASR service which is not the focus of *JustSpeak*, our hope is to collect user feedback and suggestions in the real world through channels like Play Store comments and email lists[3]. Since the first beta release of *JustSpeak* in October 2013, we have had hundreds of users. They have given us many meaningful suggestions, the overall feedback is positive among users, one of them posted appraisal that *"JustSpeak is the beginning of a fine personal assistant product"* and he is *"anxious to see how it progresses"*.

As an accessibly service, *JustSpeak* is dependent on the labeling of on-screen controls like other services such as TalkBack. Unfortunately, application developers often assign lower priority to accessibility than to other features and then simply forget providing alternative text or content descriptions to visual interface elements such as images or icons, or rush the task roughly in the end. Their carelessness often causes issues and obstacles for users of screen readers and other accessibility services. We believe that by designing and promoting *JustSpeak* and other services that benefit larger user groups, we can make application developers more aware of accessibility needs.

[2]Download *JustSpeak*, *https://play.google.com/store/apps/details?id=com.googlecode.eyesfree.justspeak*

[3] *JustSpeak* email threads, *https://groups.google.com/forum/#!searchin/eyes-free/justspeak/*

Although *JustSpeak* is implemented on Android, similar designs can be easily applied on other platforms as well. The essential system support required is a delegate layer between system and application levels which plays the same role that accessibility APIs plays in *JustSpeak*. With permissions granted by users, this layer has to be capable of invoking interface manipulation and other system functions requested by applications. It also has to be able to act as an agent that monitors interface contents and events and passes necessary information to registered applications. For operating systems on which this layer is present or can be built, the same three modules architecture can be used to create a similar universal voice control system.

FUTURE WORK

Given the feedback we received from our users, we have updated *JustSpeak* several times to add new features and fix issues. We plan to continue listening to our users and observing their voice control behaviors. In addition to maintaining and improving *JustSpeak* to increase its usability, we would like to reach out to more users by adding more language supports as well.

We are also exploring other techniques to activate *JustSpeak*, most recently some mobile devices have been equipped with always-listening hot words recognition, for example, on Motorola X and Nexus 5, users can always launch Google voice search by speaking *"OK, Google"*. We are looking into the possibility of using same technology in *JustSpeak* to enable real voice-only interaction experience.

CONCLUSION

In this paper, we have presented the system designs and use cases of *JustSpeak*, a universal voice control assistant on Android operating system. The contributions of *JustSpeak* are twofold. First, it is the first voice control application that provides enhancements to all applications running on a mobile system by synthesizing commands set from on-screen context. Secondly, it supports chaining of multiple commands in the same utterance which enables more natural and seamless interaction experience. As an application released to public, *JustSpeak* can benefit large number of users with universal eyes-free and hands-free voice control of their mobile devices. User feedback shows *JustSpeak* is welcomed by real world users. Its framework may help to shape future voice control devices.

ACKNOWLEDGMENTS

This work was supported by National Science Foundation Awards #IIS-1149709 and #IIS-1116051, and by Google.

REFERENCES

1. Apple Inc. VoiceOver. *http://www.apple.com/accessibility*
2. Apple Inc. Siri. *http://www.apple.com/ios/siri/*
3. Bigham, Jeffrey P., Craig M. Prince, and Richard E. Ladner. WebAnywhere: a screen reader on-the-go. In *Proceedings of the 2008 international cross-disciplinary conference on Web accessibility (W4A)*, pp. 73-82. ACM, 2008.
4. Google Inc. Google Now. *https://support.google.com /websearch/answer/2842392?hl=en*
5. Google Inc. YouTube. *http://www.youtube.com*
6. Google Open Source Project. TalkBack: An Open Source Screenreader For Android. *http://google-opensource.blogspot.com/2009/10/talkback-open-source-screenreader-for.html*
7. JAWS Screen Reading Software. *http://www.freedomscientific.com/products/fs/jaws-product-page.asp*
8. Kyle Alspach. Nuance: 'Influx' of Artificial Intelligence Taking Voice Recognition to Next Level. *Boston Business Journal*, October 2013.
9. Lamere Paul, Philip Kwok, Evandro Gouvea, Bhiksha Raj, Rita Singh, William Walker, Manfred Warmuth, and Peter Wolf. The CMU SPHINX-4 speech recognition system. In *IEEE Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP 2003)*, Hong Kong, pp. 2-5. 2003.
10. Lei Xin, Andrew Senior, Alexander Gruenstein, and Jeffrey Sorensen. Accurate and Compact Large Vocabulary Speech Recognition on Mobile Devices. *INTERSPEECH*, 2013.
11. Mackenzie, Ian Scott. Fitts' law as a performance model in human-computer interaction. (1992).
12. McKiel Jr, Frank A. Method and system for enabling a blind computer user to locate icons in a graphical user interface. *U.S. Patent 5,287,102*, issued February 15, 1994.
13. Povey Daniel, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann et al. The kaldi speech recognition toolkit. In *IEEE 2011 workshop on automatic speech recognition and understanding.* 2011.
14. Rabiner, Lawrence R. A tutorial on hidden Markov models and selected applications in speech recognition. In *Proceedings of the IEEE* 77.2 (1989): 257-286.
15. Raman T.V., Charles L. Chen, Tim Credo. Leveraging Android accessibility APIs to create an accessible experience. *Google I/O*, May 2011.
16. Voice Command Device. *http://en.wikipedia.org/wiki/Voice_command_device*