COSC 345 Assignment 2

Nathan Kennedy 1006384, Kat Lilly 8867318, Tobyn Packer 5003352, and Daniel Davidson 375621

May 28, 2018

Alpha Release

At this point we have an android application consisting of a main menu from which a user may chose a game to play. The only game currently working is 'Teragram', a maths game for young children. Choice of which game to play, as well as play of the math game can be done both by tapping the buttons on screen, or by voice command only.

The application uses pocketSphinx to interpret speech commands available here: https://github.com/cmusphinx/pocketsphinx

Known issues

- Currently the user can't exit from Teragram via speech commands. Also, after manually exiting from Teragram the speech recognition does not restart.
- Number input with speech commands is finicky, frustrating and slow.
- Upon first operation of the app, you will experience a crash. This is because on the first operation, pocketsphinx needs to copy our voice-recognition models into the phone's memory. This produces a crash. However, upon restarting the application, this crash isn't encountered again.

Intended improvements

Speech Recogntion - Visual Feedback

We would like to implement some kind of visual feedback telling the user which mode the speech-recognizer is in, specifically when it is in number entry mode. Currently we only have the 'What I heard' box that we are using for debugging/development and makes it obvious to us what is going on, but to a user unfamiliar with the commands, it could lead to confusion.

More robust number entry

The current way of inputing numbers using speech is finicky and prone to errors. We are unsure whether or not there is some easier way to do this, but we would like to try different methods.

Speech Recognition off-switch

We would like to implement a way of turning off speech recogntion as the game will interpret and control the game when it hears anything at all, even noises not intended to be interpreted.

Help Screen

Without this documentation, a user would have no way of knowing exactly what to say to the game to get it to do anything. Some kind of pop-up display you could call at any moment displaying the voice commands is neccessary.

User Documentation

Teragram

The game starts up at 'level 1', and you can move to the easiest level by tapping the 'too hard' button or go up levels by tapping the 'too easy' button. Automatic levelling up happens when you get ten questions right in a row, or levelling down when you get two wrong in a row.

For the unarmed

You can control the game with the following voice commands:

Main menu

- **'game one'** - unsupported
- **'game two'** - this will take you to the game Teragram.
- **'game three'** - unsupported
- **'game four'** - unsupported

Teragram

- **'harder'** - Provides a harder question.
- **'easier'** - Provides an easier question.
- **'new question'** - Provides a new question of the same type
- **'addition'** -Provides an addition question
- **'subtraction'** - Provides a subtraction question
- **'multiplication'** - Provides a multiplication question
- **'number'** - Starts Number Input mode, see below
- **'exit'** - unsupported

Number Input

- **numbers 0 9 inclusive** Appends the spoken number to the end of your answer, ie. Speaking 'one', 'two' will give you 12. Note that 0 is to be spoken 'zero'
- **'enter'** - Submits the current answer and exits Number Input mode
- **'back'** - Removes the last number from your answer
- **'clear'** - Removes the entirety of your answer
- **'cancel'** - Exits Number Input mode

When entering numbers, please wait until the number you have just spoken appears in the box before trying to input the next one. Good Luck!

What we stole

The code for the speech recognizer setup/permissions/asset copying is taken almost directly from the pocketSphinx demo application here: https://github.com/cmusphinx/pocketsphinx-android-demo

The phoneme definitions in our nktd.dic file are selectively taken from the cmudict-en-us.dict file in that same demo.

The acoustic model we use is provided by CMUSphinx at sourceforge.net

What we learned

Android is not 'just Java'

When we first began writing code for our project, we began by writing a Java application. It wasn't until we had gotten a not insignificant way into the development process that we learned that android uses different graphical libraries and not long after, that it used different audio libraries too. Everything we had written up to that point needed to be translated into android-friendly code or scrapped entirely.

Testing is hard

Our initial efforts were focused on getting an integrated testing environment up and running. This involved setting up travis CI and writing tests to ensure nothing was being broken between pushes. As it turns out, setting up a travis environment for android is not simple. There are multiple versions of the Android SDK to worry about and licence agreements and of course, gradle. We did ultimately get a travis environment set up in the end, so we do know when something is pushed that won't build.

Android Studio is hard

All of us were unfamiliar with the Android Studio IDE and all of us have struggled with it. Some of us were having trouble getting it to acknowledge that the Android SDK was in fact installed on our computers, while for some of us, code that built fine on somebodys computer would not on anothers. It seems to be a problem that occurs whenever the projects file structure was changed. Sometimes restarting Android Studio fixes the problem, sometimes explicitly forcing Android Studio to sync the project with Gradle fixes the problem and sometimes it wont. In the end we had two out of four computers building the same code, enabling us to get something done.

Adapting other people's projects is hard

When things were starting to look bleak in terms of how much we would be able to present for this deliverable, we tried to adapt open-source android games and impose our speech commands onto them. Coupled with our Android Studio file structure problems, this was a disaster. Manifests needed to be edited, other peoples code needed to be learnt, and many bugs needed to be found and fixed. We did waste plenty of time on this idea, before writing a new game from scratch that has fewer bugs in it and is more well suited to voice control.