LEARNING TO PROGRAM CONVERSATIONALLY: A CONVERSATIONAL AGENT TO FURTHER DEMOCRATIZE PROGRAMMING

J. Van Brummelen[1], C. Yeo[2], K. Weng[1]

[1] MIT (UNITED STATES)
[2] Harvard (UNITED STATES)
jess@csail.mit.edu, cyeo@college.harvard.edu, kweng@mit.edu

Computers let us solve problems: There are pre-made applications to remind us to water plants, connect us to people across the world and even help us find our way home. What if, however, you had a community-specific problem that didn't yet have a pre-made application? What if you could think of a solution to your problem, but didn't have the skills to develop it? Today, not knowing how to program computers is a serious disadvantage.

Many technologies aim to solve this problem; for example, MIT App Inventor, which empowers people as young as primary school students to develop their own mobile apps. With App Inventor, students have created apps to track school supplies, teach disinfection skills, and help reduce carbon footprints [1]. With such technology, students can move from computational thinking to computational action: They can create their own technological solutions to problems they care about [2].

Interacting graphically, as in App Inventor, is no longer the only primary mode of human-computer interaction. With the rise of conversational agents, like Amazon Alexa and Siri, many people interact using voice. Its convenience and efficiency make it a viable method for sending messages and gathering information. For similar reasons, we see voice as an effective method to lower the barrier to entry to programming and teach an even wider audience computational thinking (CT) skills. Imagine telling your computer to make your toy race around a track while it plays your favorite song—and it actually happening!

We developed a voice-based, conversational agent to do just this: empower nearly anyone to program—just by having a conversation. The system is capable of program creation, debugging and execution using four main modules: the voice user interface (VUI), natural language understanding (NLU), dialog manager (DM) and program editor. The VUI is what users interact with and speak to: it receives and transcribes users' spoken input and utters back an appropriate response determined by the DM. The NLU processes user input, extracting intent and semantic features. The DM tracks the conversation, system state, and goals. Goals represent the user intent and the actions the system must take to resolve the intent, like changing the state or asking the user for further input. Each program is represented by an object and the program editor maintains and modifies the program, according to users' descriptions.

This system has been integrated into a website that we will use to investigate research questions through user studies. First, we will determine whether using voice is beneficial to programming, which we will test by letting students compare voice- and text-based programming systems. We also plan to determine the extent to which users can learn to program using just voice, given cognitive load constraints. Lastly, through student workshops, we will investigate the extent students learn CT and artificial intelligence principles with this voice programming system.

Through workshops and user studies, we plan to improve voice-based technologies and empower nearly anyone to solve real-world problems through programming conversationally.

References:
[1] MIT App Inventor. MIT App Inventor app of the month. http://appinventor.mit.edu/explore/app-month-gallery.html, 2019. Accessed: 2019-12-05.
[2] Mike Tissenbaum, Josh Sheldon and Hal Abelson. From computational thinking to computational action. Communications of the ACM, 62(3):34–36, 2019.

Keywords: educational technology, computer science, K-12 education, conversational agents, artificial intelligence, natural language programming