Child-friendly Programming Interfaces to AI Cloud Services

Ken Kahn[1] and Niall Winters[1]

[1] Department of Education, University of Oxford,
15 Norham Gardens, Oxford OX2 6PY, UK

toontalk@gmail.com

Abstract. AI cloud services are available for speech synthesis, speech recognition, image and video recognition, text analysis, and machine learning. School students could use these services in a wide variety of programming projects including voice commands to robots, chatbots, audio games, and vision-based robotics. In doing so they may learn about perception, language, psychology, and the latest empowering technologies. A major obstacle to using these services in schools is that they are technically complex APIs beyond the ability of most school students. The challenge addressed in this paper is how to provide interfaces that are much easier to use and yet still supports most of the functionality of these AI services. We describe the addition of new programming blocks to the Snap! visual programming language [1] that provide easy-to-use interfaces to these services. We have developed new blocks for speech input and output and image recognition. Learning materials have been developed and preliminarily trialed with a small number of children.

Keywords: Visual programming, block languages, Snap!, AI services, Cloud services.

1 AI Cloud Services for Student Programmers

1.1 AI Cloud Services

Several companies are offering AI cloud services via a web connection. These include Google's machine learning services, IBM Watson cloud services, Microsoft cognitive services, and Amazon AI services. Many of these services are recognition services for providing descriptions of what is being spoken or seen. Others analyze text for content, tone, and sentiment. They all include machine learning services that find patterns in data. These are commercial services that cost a few dollars for a thousand queries. Fortunately for schools with limited budgets, free quotas are provided which allow a few hundred queries per day.

The service providers support accessing these services from many programming languages. Unfortunately, these are complex interfaces designed for use by professional

programmers. In this paper, we show how to provide easy-to-use interfaces to these services, opening up their potential to children who are learning to program.

1.2 Toward Student Projects Relying upon AI Cloud Services

Students today are often doing physical computing projects involving micro-controllers such as Raspberry Pi, Arduinos, or Micro:bits or they are programming pre-built robots. In many cases these projects could benefit significantly from the ability to recognize what is being spoken or what is in front of a camera. For example, a student could build a robot that when it hears "push the red ball" will move to the ball and push it. This could be accomplished by sending the output from a microphone to an AI cloud service, picking out the keywords in the response, then repeatedly turning and sending images from a camera to a service until the response is that a red ball is in the image and then heading in the direction the camera is facing.

There is a long tradition of children programming language-oriented programs. In the early days of Logo children programmed poetry generators, silly sentence makers, chatbots, and more [2, 3]. The appeal of these kinds of projects increases when speech input and output replaces typing and reading, an area of research that has been neglected but can now be revisited to use the analytical power of cloud-based AI services. In addition, student projects can use other AI services including sentiment analysis of what is spoken to respond in appropriate or amusing ways. This opens up the potential of AI to children in a simple and interactive manner, an emerging area of research we are exploring. Our first efforts are described below.

2 Creating Child-friendly Programming Interfaces

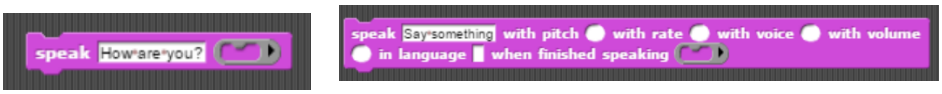
Our goal is not to create a new programming environment for children, but instead to enhance existing ones. We have added speech input and output to ToonTalk Reborn [4] and to Snap! [1]. We have also added image recognition to Snap!. This paper reports on our Snap! efforts.

2.1 Why was Snap! chosen?

Snap! is a superset of the very popular children's blocks-based programming language Scratch [5]. It is well-suited to our efforts because (1) it is a powerful language that supports first-class data structures and functions; (2) it is easy to define new blocks using JavaScript without touching the source code; (3) it runs in every modern browser; (4) and there are versions that connect to Arduinos and the Raspberry Pi.

2.2 Speech synthesis

All the popular browsers except for Internet Explorer support the Speech Synthesis API. The API utters the provided text with control for the pitch, rate, volume, language, and voice. The first version of the speak command took a text argument and an optional function called when the speaking finishes while the second version exposed nearly all the functionality of the Speech Synthesis API:



While this looks more difficult to use, all but the first parameter is optional and can be ignored. As discussed later students were clearly amused by entering different values for the pitch, rate, and voice.

2.3 Speech recognition

While speech recognition is part of the Web Speech API, as of this writing only Chrome and Opera support it. However, an AI cloud service for speech recognition is available from Google, Microsoft, and IBM. The Snap! speech recognition block we designed has a success continuation and an optional error continuation.

The asynchronous nature of recognition services forces a reliance upon continuations. Continuations are ideal from a technical point of view; however we are concerned that student programmers may find them difficult. To address this, we implemented a block using the 'listen' block that supports event broadcasting and a global variable. This complexity is hidden (but available for the ambitious students) so that one needs only to call 'listen' and then receive broadcasts when something was heard and read a global variable containing the last thing spoken. For example, this program repeats what was spoken, prefaced with "I think I heard you say":

2.4 Image recognition

There are no standard APIs for image recognition so the block we implemented supports the Google, IBM, and Microsoft APIs. A block is also needed to setup the camera. The image recognition block has a parameter specifying which cloud provider, a continuation that will receive a description of the image, and a flag as to whether the image should be displayed.

3 User Testing

As of this writing the speech input and output blocks have been tested with about 25 undergraduate students and 6 children (aged 7 to 13). This preliminary testing has been very encouraging. The new Snap! blocks were easily understood and the users indicated that they enjoyed adding speech to their programming projects. One student asked for a way to respond to nothing being said. A good suggestion we are exploring. Some students discovered that it is best to program a final 'else' clause for speech recognition to provide user feedback instead of ignoring those utterances that aren't understood.

4 Discussion and Future Directions

From the early days of Logo research [2, 3] there was interest in supporting children in creating artificial intelligence programs. The decision making, perception, learning, and natural language understanding in these programs was necessarily very simple. In the process of programming AI system one is forced to reflect on one's own thinking processes. This provided a very good fit for the constructionist ideas of learning through construction and reflection.

Today's AI cloud services provide a new opportunity to support a new class of student AI projects – those that rely upon a state-of-the-art AI "subroutines". Children can design and build impressive intelligent artefacts by composing and customizing components provided by world-class AI teams. Among the reasons for doing this are that students (1) may become better motivated and empowered to produce very capable artefacts; (2) may learn about perception, reasoning, psychology, and animal behavior in the process of building perceptive robots and apps; (3) may learn about cloud services, artificial intelligence, and other advanced technologies; and (4) may reflect more deeply upon their own abilities to hear, see, and respond appropriately.

There are many open issues that need further research: (1) How to incorporate recognition confidence scores provided by the AI services? (2) How to provide user control over the kinds of image recognition desired and the format of the responses? (3) Should broadcasting alternatives of all the continuation-based blocks be provided? (4) How to make the blocks multi-lingual? (5) What additional AI services should be added to the current collection? (6) How different would child-friendly interfaces to AI cloud services be when integrated with other programming systems such as ToonTalk [4]. These future developments will proceed iteratively with user testing to provide insights into the children's perception of AI. With much discussion today focused how AI will "put people out of a job", ways in which children can be supported to program AI services promotes learning and understanding of AI's potential and the skills by which to use it in new, creative and effective ways.

These programs are available at tinyurl.com/ai-snap-demos. This research was supported by the eCraft2Learn project [6] funded by the European Union's Horizon 2020 Coordination & Research and Innovation Action under Grant Agreement No 731345.

1. Harvey, B., Mönig, J.: Bringing "No Ceiling" to Scratch: Can One Language Serve Kids and Computer Scientists? In: Constructionism, Paris, France (2010).
2. Papert, S., Solomon, C.: Twenty Things to Do with a Computer, MIT AI Lab, http://hdl.handle.net/1721.1/5836, (1971).
3. Kahn, K.: A Logo natural language system. Technical report, MIT AI Lab, LOGO Working Paper 46 (1975).
4. Kahn, K.: TOONTALK REBORN - Re-implementing and re-conceptualising ToonTalk for the Web. In: Constructionism, Vienna, Austria (2014).
5. Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B. and Kafai, Y.: Scratch: programming for all. Communications of the ACM 52(11), 60-67 (2009). DOI=http://dx.doi.org/10.1145/1592761.1592779
6. http://www.project.ecraft2learn.eu/