Conversational Artificial Intelligence Development Tools for K-12 Education

Jessica Van Brummelen
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139
jess@csail.mit.edu

Abstract

Artificial intelligence (AI) education is becoming increasingly important as the adoption of AI technology, including conversational agents such as Amazon Alexa, Siri, and the Google Assistant, increases. Current educational and programming tools enable non-programmers to develop simple conversational agents, or advanced programmers to develop complex agents. However, to the author's knowledge, there are no tools for non- or novice programmers to develop conversational agents for the purpose of learning AI and programming skills. This paper describes AI curriculum that includes content about conversational agents, machine learning (ML), and AI ethics, as well as a blocks-based conversational AI interface developed within MIT App Inventor. During a series of six workshops, students used this interface to develop conversational agents for social good, including a memory aide, math tutor, speech visualizer, and recycling assistant. In this paper, I present **(1) the blocks-based interface**, **(2) the conversational AI curriculum**, **(3) how conversational AI directly relates to computational thinking skills**, and **(4) results from an initial small-scale study**. The results show that through the curriculum and using the blocks-based conversational AI interface, students **learned AI and ML concepts**, **programming skills**, and **to develop conversational agents for social good**.

Introduction and Related Work

The importance of artificial intelligence (AI) education is becoming more evident with AI's increasing ubiquity. For instance, in one study where children and adults observed a robotic toy navigating a maze, the majority of participants indicated they thought the toy was smarter than they (Druga et al. 2018). In addition to the relevance of teaching about AI technology, it is important to emphasize AI's social implications. For instance, conversational agents may be used *positively* to help drivers navigate the bewildering streets of Boston, or *negatively* to phish for confidential information. The need for AI education is especially apparent considering AI democratization tools, like Google's AIY, Anki's programmable Cozmo robots, and Scratch's Cognimates extensions (Touretzky et al. 2019), which enable people without extensive training in computer science to meaningfully participate in AI development.

Other tools to democratize and teach AI include *MIT App Inventor extensions*, such as the image recognition, speech processing and text analysis extensions (Zhu 2019); *PopBots*, a platform for preschoolers to train and interact with AI (Williams, Park, and Breazeal 2019); and *Machine Learning for Kids*, a platform to generate machine learning (ML) models and develop web or mobile apps using *Scratch* or *MIT App Inventor* (Lane 2018). Each of these tools focus on teaching essential ML and AI concepts through empowering learners to develop AI-enabled projects.

Nonetheless, there are relatively few AI democratization tools that address conversational AI (the ability of a machine to interact with humans using natural language), which is rapidly becoming ubiquitous with devices like the Amazon Echo (as well as becoming a hot area of research). Furthermore, the few tools addressing this area are typically created for one of two purposes: (1) to enable non-programmers to create standardized apps with little programming knowledge (e.g., fill-in-the-blank Alexa Skill Blueprints (Amazon 2019)) or (2) to enable skilled developers to create complex conversational agents (e.g., Google Actions Console (Google 2018)). These tools do not provide scaffolding for the beginner- or non-programmer to learn AI development skills, or purposeful educational opportunities about the implications of conversational AI.

To address this lack of educational, conversational AI development tools, I created a blocks-based programming interface in MIT App Inventor, as shown in Figure 1. This visual coding platform enables a range of development (from simple to highly complex apps), simplifies the programming process, and promotes computational thinking (CT) skills. More specifically, such *blocks-based* interfaces lower the barrier of entry to programming, enable quick prototyping, and encourage learning gains in computer science (Weintrop and Wilensky 2017). Furthermore, conversational AI programming involves complex technical terminology (e.g., *intents*, *slots*, *long short-term memory (LSTM) networks*, etc.); thus, to enable students to learn such complex concepts, it is helpful to ease learning in other areas (e.g., through a straightforward *blocks-based* development environment).

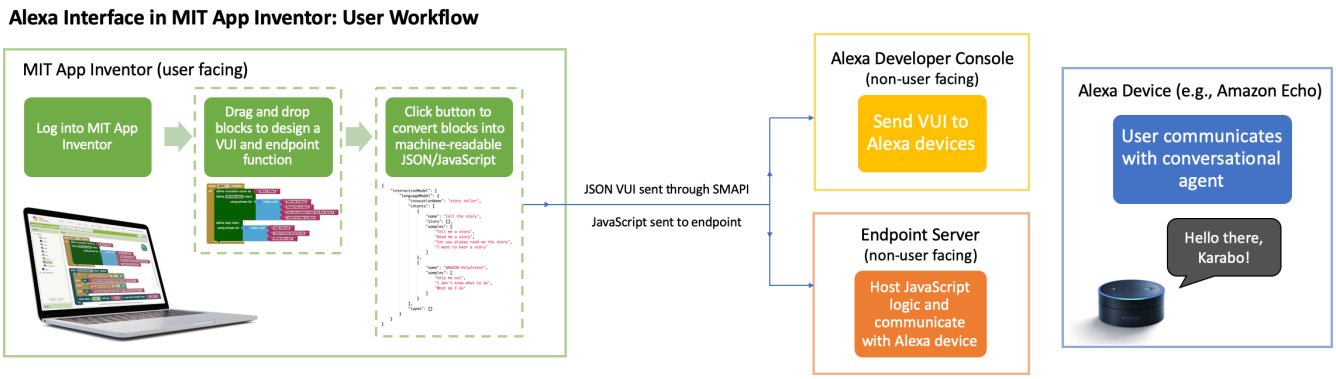

Figure 1: User workflow to create a conversational AI agent. The user first implements the Voice User Interface (VUI) and endpoint function using a blocks-based interface. The blocks are converted to JSON and JavaScript, which define the agent's functionality on Alexa devices.

Additionally, the platform provides students with high-agency, project-based learning opportunities. This is in contrast to tools such as *Zhorai* and *PopBots*, which are aimed at a younger audience, and are less focused on project development, but rather on exposure through conversational agent interaction (Van Brummelen, Lukin, and Lin 2019; Williams, Park, and Breazeal 2019). Research suggests that curricula in which students develop STEM projects increases students' creative thinking skills, STEM skills efficacy, and STEM career aspirations (Lestari, Sarwi, and Sumarti 2018; Beier et al. 2019); thus, for the purpose of STEM and CT skill development, I created a project-based curriculum. This curriculum involves brainstorming real-world problem solutions, generating designs, and developing conversational agents, as described in the *Curriculum* section. I investigate the effectiveness of this curriculum and the conversational AI interface through a workshop-based, small-scale study, as outlined in the *Results* section.

Motivational Scenario

To provide a basis for understanding (1) the components of the conversational AI interface, (2) how someone would use the interface, and (3) the capabilities and limitations of the system, I present a scenario about how "Sheila" created a conversational AI app for her cousin "Jaidon". Although the scenario is fictional, a version of *Sheila's Storybook App* was implemented using the interface and presented to students in the workshops. Furthermore, actual applications students developed are discussed in the *Results* section.

Sheila's Storybook App

Sheila, a seventh grade student, loves stories. When she was younger, she imagined jumping into the pages of her storybook and interacting with the characters. During a computer lesson, she heard about MIT App Inventor's conversational AI interface and had a brilliant idea: to create a talking storybook. Sheila would create the storybook app using MIT App Inventor, run it on her tablet, and enable conversation using the Alexa app. The storybook would have the following main features:

- You could swipe through "pages" of the storybook while reading and viewing illustrations on-screen
- You could ask Alexa about the characters, setting, and narrative (e.g., Figure 2)
- You could ask Alexa to read you the story, and as Alexa reads, the sentence on the app's page would be highlighted
- You could have "conversations" with the storybook characters, and when you ask a character a question, a response would be automatically generated

Figure 2: Speaking with Alexa contextually with Sheila's storybook. Modified from (Van Brummelen 2018).

To implement the storybook app, Sheila first uploaded illustrations to MIT App Inventor and implemented page-flipping functionality by creating *events* in the blocks-based interface. When the "next" button was pressed, a counter would increase, and the next illustration would pop up on screen. The opposite event would occur when the "previous" button was pressed. After creating the mobile app (as shown in Figure 3), Sheila moved onto the conversational AI portion of the app.

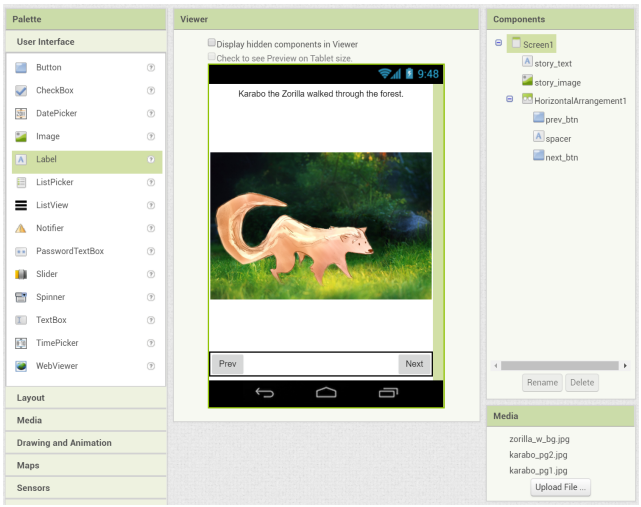

Figure 3: The storybook mobile app being developed on the MIT App Inventor website.

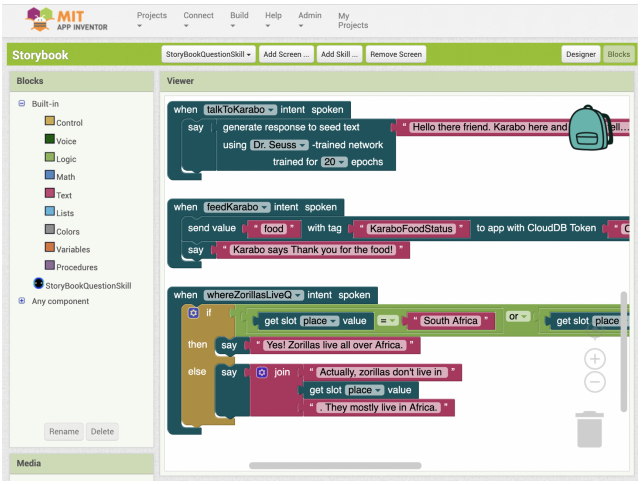

Figure 4: Sheila's storybook endpoint function blocks in MIT App Inventor. Notice the *when intent spoken*, *say*, and *generate text* blocks.

Sheila added a new Alexa Skill to the project by clicking the *Add Skill* button. This brought her to a page where she could drag-and-drop and connect blocks together to create a conversational agent. First, she dragged out a *define intent* block so that the Voice User Interface (VUI) would recognize when someone said, "Tell me a story". She wanted the agent to respond by telling a story about zorillas (a little-known animal that Sheila absolutely *loved!*), so she also dragged out a *when intent spoken* block. She connected the *when intent spoken* block to a *say* block containing a *text* block with the first line of the story.

Sheila also wanted people to be able to speak with the main character, Karabo the Zorilla. She didn't want to write out all the possible answers to people's questions though (that would take *forever*), so she decided to use a *generate text* block to generate Karabo's answers instead. According to her seventh grade teacher, this block used *machine learning* to generate sentences sounding kind of like the stories in the block's drop-down menu. Sheila imagined Karabo speaking kind of like Dr. Seuss, so she chose that one from the menu.

After adding some additional functionality using blocks, as shown in Figure 4, Sheila sent the Alexa Skill and mobile app to her cousin Jaidon. Jaidon downloaded the app and started flipping through the pages, talking to the storybook as he went along. Jaidon was thrilled that he could listen to Alexa read him the story, especially since he didn't know how to read himself. He also had a blast asking Karabo questions and hearing the infinite different ways Karabo would respond (despite the sentences not always being logical). He laughed when Karabo said, "Little cat in the hat, they can not eat them in your snow". It sounded quite like one of his favorite Seuss stories. Hearing Jaidon's laughter meant the world to Sheila. In her mind, an app that allowed her to connect with her cousin thousands of miles away was a huge success.

Curriculum and Workshop Series Overview

The curriculum for the conversational interface specifically focuses on teaching the eighteen CT and AI concepts, practices, and perspectives presented in (Brennan and Resnick 2012) and (Van Brummelen, Shen, and Patton 2019). Tables 1, 2, and 3 elaborate on how each CT/AI dimension corresponds to conversational AI. We taught these dimensions during a series of six workshops, based on the success of the workshops presented in (Lao 2017).

In addition to CT/AI dimensions, the curriculum teaches concepts specific to conversational AI. Conversational AI can be difficult to begin learning due to the technical jargon and unique fundamental concepts. For instance, *slot filling* is the fundamental conversational AI concept of acquiring specific information from the user that is required for the agent to perform some action (e.g., acquiring the type of food a user is craving before ordering it). See tables 1, 2, and 3, as well as (Van Brummelen 2019) for further descriptions of conversational AI concepts in the context of the curriculum.

The full curriculum can be found in (Van Brummelen 2019), and was implemented in a workshop series running from Feb. 23 to Apr. 6, 2019 through MIT's High School Study Program. Students learned about computer programming, conversational AI concepts, and how to develop conversational agents. The series culminated in a final project, in which students developed conversational agents to address real-world problems. The contents of each workshop is outlined in Table 4.

Design and Technical Implementation

The main goals of the conversational AI interface's design were to empower students with little or no programming experience to **(1) learn CT skills**, **(2) learn conversational AI concepts**, and **(3) develop conversational AI applications**. Figure 5 illustrates examples of how the interface implements these three goals.

Table 1: Correspondence between the CT/AI concepts, and conversational AI.

Computational Concept	Relation to Conversational AI
Sequences	Sequences are taught in the context of conversational turn-taking, word and sentence order, and LSTM networks.
Loops	When questions are repeated in conversation because someone misheard, the repetition can be represented as a loop.
Events	Conversational agents take advantage of events when they hear "invocation names" and *begin* listening.
Parallelism	Blocks in the conversational AI interface, as well as the regular MIT App Inventor interface can be executed in parallel. For example, while the user is speaking with a conversational agent, other events can occur on-screen in the connected app.
Conditionals	To decide what happens in a conversation, conditionals can be used. For example, *if* Alexa receives the response, "dogs", to the question, "What's your favorite animal?", *then* she could respond with "I agree!"
Operators	The "equal to" operator may be used to check the equivalency of a user's response to another string.
Data	To contextualize conversation, one can share data between the agent and connected app.
Classification	During speech recognition, conversational agents classify parts of waveforms into phonemes, then words, and ultimately utterances.
Prediction	When using the *generate text* block in the conversational AI interface, a neural network iteratively predicts a best next letter until a complete sentence is formed.
Generation	The *generate text* block generates sentences using pre-trained LSTM networks. This enables Alexa to respond with unique sentences.

Table 2: Implementation of the computational and AI practices in the conversational AI curriculum.

Computational Practice	Curriculum implementation
Being incremental and iterative	Students are encouraged to implement small changes to their conversational AI programs, test, and repeat.
Testing and debugging	Students are encouraged to test often and try as many possible user inputs as possible.
Reusing and remixing	Students develop a standard conversational AI program using a tutorial. Afterwards, they are encouraged to modify and personalize it.
Abstracting and modularizing	Students modularize their code through function development. For example, they may create a function to decide whether a comment is positive or negative, and use it multiple times.
Training, testing, and validating	Students learn about ML and generative AI through the *generate text* block. This block has a drop-down menu that enables students to vary the time spent training, and the training corpora of an LSTM network. Students can then test the models, observe the difference in output with various training times and corpora, and validate the effectiveness of the implementation.

Table 3: Correspondence between the CT/AI perspectives and the conversational AI curriculum.

Computational Perspective	Implementation in proposed research
Expressing	Students can express themselves through developing creative, unique conversational agents.
Connecting	Voiced conversation is conspicuous. Thus, students inevitably hear other students' agents and connect with each other, sharing newly discovered learnings.
Questioning	Even with instruction, understanding AI is difficult. The curriculum includes discussion questions, such as "How do ML and symbolic AI differ?". Students are encouraged to ask questions about conversational AI.
Evaluating	It is important to evaluate technology and ask questions such as, "Does this program accomplish the task I intended it to accomplish?", and "Is this technology good for society?". Students are encouraged to evaluate their projects' quality, capabilities, limitations, and potential outcomes.

Despite the simple-looking interface, the backend is complex. For instance, the *generate text* block (highlighted in blue) abstracts away API calls to a server, twenty-eight LSTM networks pretrained for various lengths of time and on various corpora, and callbacks to the Alexa Developer Console. The "send to app" and "get from app" blocks abstract away a Redis database, and the remaining "Voice" blocks used to create conversational agents abstract away

Table 4: Workshop series overview. Each workshop consisted of a short conversational AI lecture, as well as time to work on unplugged activities or developing applications.

Workshop	Overview
1	• Introduce programming and mobile app development with MIT App Inventor. • Teach students how to program in MIT App Inventor. (See tutorial list in (Van Brummelen 2019).) • Complete pre-questionnaire assessment. (See questionnaires in (Van Brummelen 2019).)
2	• Introduce conversational AI and Alexa skills. • Explain basic AI concepts, such as the difference between rule-based AI and ML, and terminology, such as neural networks, training, and testing. • Explain basic conversational AI terminology, such as *VUI*, *invocation name*, *utterances*, and other terms. • Complete conversational agent unplugged activity. (See activity in (Van Brummelen 2019).) • Introduce the conversational AI interface in MIT App Inventor through a storybook skill. (See skill in (Van Brummelen 2019).) • Provide students with worksheets related to the "Talking to a Storybook" skill.
3	• Review the "Talking to a Storybook" solutions. • Give time for the students to remix the storybook skill and create their own skills.
4	• Introduce the conversational AI design project. • Brainstorm project ideas and create a project plan. • Provide time for students to start programming their projects.
5	• Briefly review conversational AI topics, including ML and rule-based AI. • Provide students with time to complete final projects and prepare presentations.
6	• Provide time for final touches and presentations. • Complete post-questionnaire assessment.

API calls to Amazon Web Services and the Alexa Developer Console, as well as communication between Amazon and Alexa devices, as shown in the architecture diagram in Figure 6. This abstraction enables students to easily create ML-powered conversational agents without worrying about JSON formatting, the status of various web servers, or even simple syntax errors, like missing semicolons.

The main elements of the system are shown in the user workflow diagram in Figure 1. Namely, I implemented (1) a *designer page* in MIT App Inventor where the user can create and send the Alexa skill to Amazon, (2) a *blocks page* where the user can program the Alexa skill, (3) fifteen new conversational AI ("Voice") blocks used to program Alexa skills, (4) a web server that the "generate text" block uses to access pretrained LSTM networks, and (5) a communication interface between Amazon's Alexa Developer services and MIT App Inventor. This empowers learners to create Alexa Skills, or conversational AI agents like "Sheila" did in the *Storybook App* section.

In summary, the interface enables learners to develop agents that can **converse with the user and respond to utterances**, **share data with mobile phone applications developed in MIT App Inventor**, and **generate unique responses using ML**. Further information about the implementation can be found in (Van Brummelen 2019).

Field Study Results

To gather information about the effectiveness of the conversational AI interface and curriculum to teach students AI concepts, how to program conversational agents, and about the capabilities, limitations and implications of conversational AI, we gathered information through a small-scale, workshop-based study with seven participants. Before and after completing the workshop series, high school students from the Boston area filled out a questionnaire about their previous experience coding and interacting with conversational agents, understanding of conversational AI, and other related topics.

The questionnaires consisted of a series of short answer and Likert scale questions. The short answer question results are discussed in the next section and the Likert scale results are shown in Figure 7. Four of the five Likert questions were on both the pre- and post-questionnaire. A final fifth question was only on the post-questionnaire. These questions asked students to rate the following statements on a scale from one to five (Strongly Disagree to Strongly Agree):

1. I have interacted with conversational agents.
2. I understand how conversational agents decide what to say.
3. I feel comfortable making apps that interact with conversational agents.

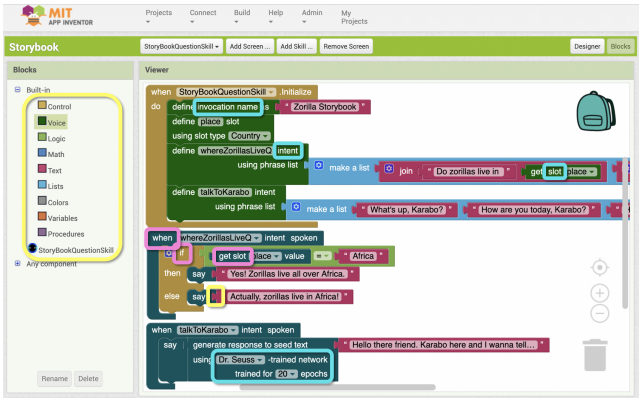

Figure 5: The conversational interface highlighted according to design goals. *Pink:* The pink boxes highlight the interface's attention to CT skills, including events (*when*), conditionals (*if*), and data (*get slot*). *Blue:* The blue boxes highlight conversational AI concepts, including *invocation name*, *intent* and *slot*, as well as a "generate text" block containing LSTM neural networks. *Yellow:* The yellow boxes illustrate the learnability of the interface. The leftmost highlight shows the "drawers" where blocks can be easily dragged-and-dropped into the interface, and the smaller yellow box shows how blocks-based coding can prevent syntax errors, as only certain blocks (e.g., "say" and "text" blocks) can be connected to each other.

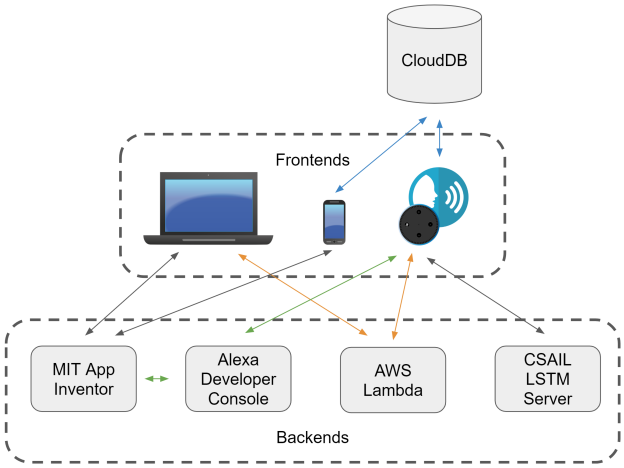

Figure 6: The architecture for the conversational AI interface. This is discussed in more detail in (Van Brummelen 2019).

4. I can think of ways that conversational agents can solve problems in my everyday life.
5. My understanding of conversational agents improved through these workshops.

As shown in Figure 7, the minimum values as well as the means of the responses increased or remained the same from the pre-questionnaire to the post-questionnaire. This suggests that students' self-efficacy in developing, solving problems using, and understanding conversational AI increased as a result of the workshops. All students either agreed or strongly agreed with the final question, further supporting that their understanding of conversational agents improved.

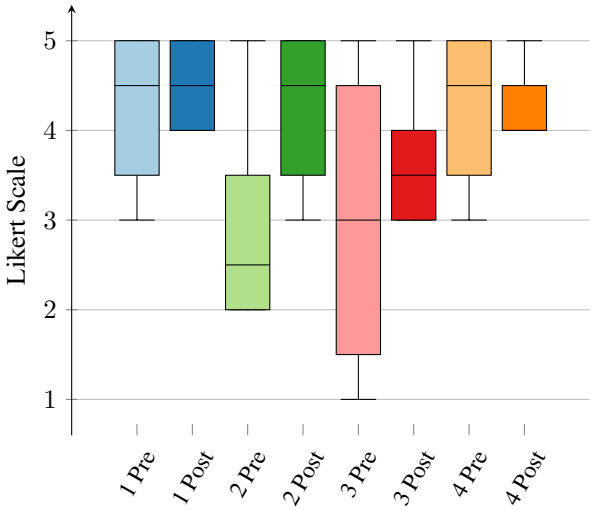

Figure 7: The results of the pre- and post-questionnaires. The darker colored boxes are the post-questionnaire results. Notice that the minimum values as well as the means of the responses increased or remained the same from the pre-questionnaire to the post-questionnaire. (Note that one student entered strongly agree (5) for every answer in the pre-questionnaire, despite his/her short answer responses not reflecting the same sentiment, and since there was a small sample size, this significantly skewed the pre-questionnaire results upwards.)

Additionally, students developed conversational agent projects, which were insightful in terms of student learning outcomes, future conversational AI applications, and future interfaces/curricula. The projects were as follows:

1. **Memory Helper: Synonym Finder**
   - Helps people remember forgotten words
   - Example utterance: "Tell synonym finder the word is a fruit that grows in an orchard"
   - Example response: "Sounds like you might be thinking of the word, 'apple'"
   - See example dialog from this project in Figure 8
2. **Can I recycle this?**
   - A tool to help people sort recyclables (or non-recyclables) correctly
   - Example utterance: "Can I recycle this plastic box?"
   - Example response: "Can you find a recycling symbol on the box?" (If yes) "What's its number?"
3. **Alexa Speech to Text**
   - Enables those with hearing loss to read Alexa's speech
   - Example utterance: "Tell me a random fact"
   - Example response: "Crickets have ears on their legs"

  - The response is displayed on a mobile phone app

4. **Math Teacher / Calculator**
   - Helps students with math
   - Example utterance: “What is three times five?”
   - Example response: “Three multiplied by five is the same thing as three groups of five items, which is fifteen in total.”
5. **Cipher/Decoding Tool**
   - Decodes hidden messages
   - Example utterance: “Decode ’KHOOR ZRUOG’ using the Caesar cipher”
   - Example response: “Using the Caesar cipher, ’KHOOR ZRUOG’ is ’HELLO WORLD’

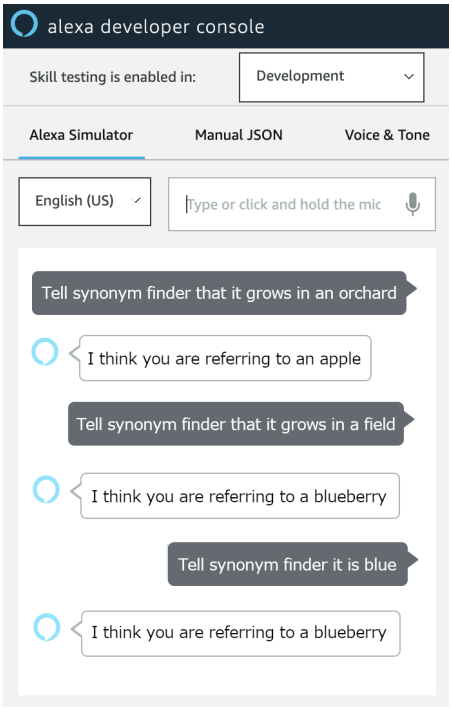

Figure 8: Example dialog from the *Synonym Finder* final project. This skill helped remind people of words they had forgotten.

Due to time constraints, not all projects were completed. With an additional workshop, or longer workshops, students likely would have completed more robust applications. Nonetheless, students generally responded positively when asked about how they felt about their projects. For example, one student stated, “I feel good about [my project]. I think it could actually help people and be put out into the world, if we put in some more work and make it function for more [cases].” Furthermore, as shown by the questionnaire results, students learned valuable AI concepts through the development process and workshops.

Discussion and Conclusions

Through analyzing the data gathered and reflecting on workshop experiences, a number of main themes, considerations, and reflections emerged. This includes the effectiveness of the blocks-based interface and project-based curriculum to empower students to learn difficult concepts, and recommendations for future curricula.

Project-based learning curricula can enable high school students to learn about complex topics such as AI. Students learned conversational AI concepts, such as the difference between ML and rule-based AI, conversational AI terminology and the capabilities, limitations, and implications of conversational agents, through the workshop curriculum. For instance, one student described ML as “morph[ing] over time in response to input” and rule-based AI as “always respond[ing] the same way to the same stimulus or intent”. Another student stated that ML is “more prone to make mistakes” whereas rule-based AI “has limitations towards how much it can do with the same amount of time spent [programming each rule]”. Students also identified privacy and security implications of conversational agents, For instance, one student stated, “They are mostly safe, but the danger can come from data collection or spying”. Another student stated “I think they are safe to use but they can become dangerous quickly because others could be listening to you and could steal your personal information”. Through the workshops, students evidently not only contemplated how to develop interesting applications, but also evaluated the safety implications of such technology.

Blocks-based interfaces enable complex learning through low barriers to entry. By leveraging MIT App Inventor’s low barrier to entry for programming, the conversational AI interface enabled students as young as ninth grade to develop conversational agents and utilize complex ML algorithms. Futhermore, the *generate text* block provided a highly abstracted, low-barrier-to-entry way to implement ML in mobile apps and conversational agents. By experimenting with LSTM models pre-trained for different number of epochs and on various input corpora, students can learn how training a ML model in different ways affects the model’s output.

Students found conversational AI concepts challenging; however, they were able develop complex, purposeful conversational AI agent projects. At first, students found the workshop content and developing conversational agents challenging. For instance, one student mentioned, “It was at first difficult to understand the way in which intents are defined”. Another student stated, “It was hard to use both the app part and the Alexa part, but I figured it out after some thinking.” Despite these difficulties, through the curriculum and blocks-based coding tools, students were able to develop conversational AI projects and explain ML concepts. This also suggests this grade level (high school) is a good fit for conversational AI curriculum.

Students were optimistic about the future of conversational AI. Through the questionnaire responses, it was clear that students could think of interesting, positive applications for conversational AI, and thought the technology would improve with time. For example, students described

conversational agents as "capable of solv[ing] problems and help[ing] people", "[able to] read things for people, suggest words, inform people, and many other things" and "definitely improve over time". Students also shared ideas for future agents, including a "tool that CADs what you explain to it by using what you say to draw images" to a "language translation [tool]".

Students' conversational AI projects were positive and purposeful. From a memory aid to a recycling management tool, students' projects addressed problems they saw in the world. The curriculum emphasized the importance of AI for social good, and as evidenced by the workshops, students were passionate about technology development for a better world.

Students found it exciting to be the first users of the interface, despite encountering bugs. Despite the interface being iteratively updated during the workshop series, students were generally forgiving of the interface. For instance, one student stated, "I did not expect that I would be one of the first people to use it, so that was very cool — to be able to use the beta". Students also noticed that it was improving throughout the series, and that their feedback was being implemented. Based on this feedback, I would encourage other researchers interested in developing educational tools to do user testing early, even if it is still under development.

Students would have appreciated more time to develop projects. Students' comments on the questionnaire indicated they were proud of their project development, and would have been interested in developing them more fully. For example, one student stated, "I am happy with the turnout and what I have learned. Hopefully, I will have the ability to extend my projects further in the future." Furthermore, some in-class activities could not be completed on time; thus, I would recommend extending the class time allotted or creating an additional workshop for project development.

Future studies may include comparisons to text-based and voice-based interfaces. The small-scale study presented here provided insight into developing a learnable, usable visual interface for conversational AI development. I plan to complete a larger-scale study to gain additional insight into students' CT/AI learnings. I also plan to develop a naturalistic language, voice-based programming tool and compare the effectiveness of text-, voice-, and blocks-based programming interfaces for AI pedagogy.

Through this research, I have had the opportunity to engage with students who are enthusiastically optimistic about the future of conversational AI. With further studies, additional AI democratization tools, and refined workshop curriculum, students will be able to develop even more positive, socially useful technologies for a better future.

Acknowledgments

I would like to thank Tommy Heng, Atsu (CC) Manigar, Terryn Brunelle, and Julia Gonik, who helped develop the interface as well as organize the workshops; Evan Patton and Jeffrey Schiller, who helped with development; Hal Abelson, who supervised this work and brought MIT App Inventor into being; and the rest of the MIT App Inventor staff (past and present). This research was supported by an Alexa Graduate Fellowship.

References

Amazon. 2019. Amazon Alexa skill blueprints. https://blueprints.amazon.com/, Last accessed on 2019-06-17.

Beier, M. E.; Kim, M. H.; Saterbak, A.; Leautaud, V.; Bishnoi, S.; and Gilberto, J. M. 2019. The effect of authentic project-based learning on attitudes and career aspirations in stem. *Journal of Research in Science Teaching* 56(1):3–23.

Brennan, K., and Resnick, M. 2012. New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual American Educational Research Association meeting, Vancouver, Canada*, volume 1, 25.

Druga, S.; Williams, R.; Park, H. W.; and Breazeal, C. 2018. How smart are the smart toys?: children and parents' agent interaction and intelligence attribution. In *Proceedings of the 17th ACM Conference on Interaction Design and Children*, 231–240. ACM.

Google. 2018. Actions on google. https://console.actions.google.com/, Last accessed on 2019-06-17.

Lane, D. 2018. Machine learning for kids.

Lao, N. 2017. Developing cloud and shared data capabilities to support primary school students in creating mobile applications that affect their communities. Master's thesis, Massachusetts Institute of Technology.

Lestari, T. P.; Sarwi, S.; and Sumarti, S. S. 2018. Stem-based project based learning model to increase science process and creative thinking skills of 5th grade. *Journal of Primary Education* 7(1):18–24.

Touretzky, D.; Gardner-McCune, C.; Martin, F.; and Seehorn, D. 2019. Envisioning AI for K-12: What should every child know about AI? In *The Thirty-Third AAAI Conference on Artificial Intelligence*, 9795–9799.

Van Brummelen, J.; Lukin, G.; and Lin, P. 2019. Zhorai: A conversational agent to teach machine learning concepts to elementary school students. https://zhorai.csail.mit.edu/, Last accessed on 2019-08-16.

Van Brummelen, J.; Shen, J. H.; and Patton, E. W. 2019. The popstar, the poet, and the grinch: Relating artificial intelligence to the computational thinking framework with block-based coding. In *Proceedings of International Conference on Computational Thinking Education 2019*, 160–161.

Van Brummelen, V. 2018. The X-mazing Zorilla.

Van Brummelen, J. 2019. Tools to create and democratize conversational artificial intelligence. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA.

Weintrop, D., and Wilensky, U. 2017. Comparing block-based and text-based programming in high school computer science classrooms. *ACM Transactions on Computing Education (TOCE)* 18(1):3.

Williams, R.; Park, H. W.; and Breazeal, C. 2019. A is for artificial intelligence: The impact of artificial intelligence activities on young children's perceptions of robots. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 447. ACM.

Zhu, K. 2019. An educational approach to machine learning with mobile applications. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA.