Designing a Conversational Interface for a Multimodal Smartphone Programming-by-Demonstration Agent

Toby Jia-Jun Li
Carnegie Mellon University
Pittsburgh, PA
tobyli@cs.cmu.edu

Brad A. Myers
Carnegie Mellon University
Pittsburgh, PA
bam@cs.cmu.edu

Amos Azaria
Computer Science Department
Ariel University, Israel
amos.azaria@ariel.ac.il

Igor Labutov
Carnegie Mellon University
Pittsburgh, PA
ilabutov@cs.cmu.edu

Alexander I. Rudnicky
Carnegie Mellon University
Pittsburgh, PA
air@cs.cmu.edu

Tom M. Mitchell
Carnegie Mellon University
Pittsburgh, PA
tom.mitchell@cs.cmu.edu

Conversational UX Design CHI 2017 Workshop, May 6, 2017, Denver, CO, USA.

Abstract

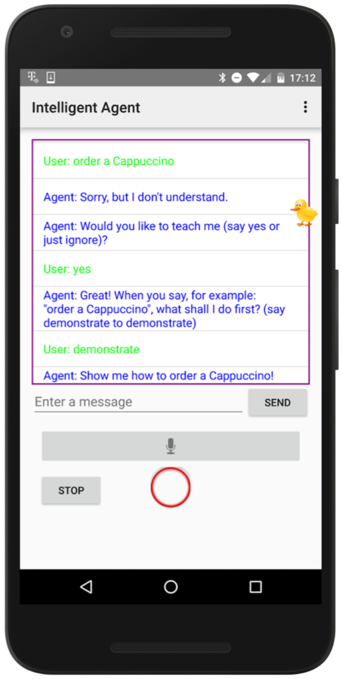
In this position paper, we first summarize our work on designing the conversational interface for SUGILITE – a multimodal programming by demonstration system that enables a virtual agent to learn how to handle out-of-domain commands and perform the tasks using available third-party mobile apps in task-oriented dialogs from the user's demonstrations. We then discuss our planned future work on enabling the end users to create more useful and usable automations for the virtual agent by supporting the users to verbally describe interface operations, to narrate ambiguous steps during demonstrations, and to specify conditionals, triggers, parameters and error handling behaviors through a conversational interface.

Author Keywords

Multimodal interaction; conversational interface; programming by demonstration; end user programming

ACM Classification Keywords

H.5.2. Information interfaces and presentation (e.g., HCI): User Interfaces: Interaction styles.



User: order a Cappuccino

Agent: Sorry, but I don't understand.

Agent: Would you like to teach me (say yes or just ignore)?

User: yes

Agent: Great! When you say, for example: "order a Cappuccino", what shall I do first? (say demonstrate to demonstrate)

User: demonstrate

Agent: Show me how to order a Cappuccino!

Figure 1: The conversational interface of SUGILITE, showing the conversation between the user and the virtual agent on demonstrating a new task "order a cappuccino"

[1]Video Link: https://www.youtube.com/watch?v=IocwhPwy5N4

Introduction

Existing conversational intelligent agents like Siri, Google Assistant, Alexa and Cortana can perform various tasks, including device control, communication, web search and calendar management. However, these agents have limited functionality. They can only invoke built-in apps (e.g., Phone, Calendar, Music, etc.) and a few integrated external apps and web services (e.g., Search, Weather, Wikipedia) and cannot perform the tasks if the user gives a command that requires the use of an unsupported third-party service. In the conversational user experience with the current agents, for a user utterance that does not match any of the system's known types, the agents often simply give an error response (e.g., "*I can't do that*", "*I don't understand that*") or perform a general web search for the user's command, which is not very helpful.

To address this limitation, we are designing and implementing SUGILITE [3], a multimodal programming by demonstration (PBD) system that enables end users to program arbitrary smartphone tasks by combining the demonstrations made by directly manipulating the regular graphical user interface (GUI) of smartphone apps, along with verbal instructions from the users.

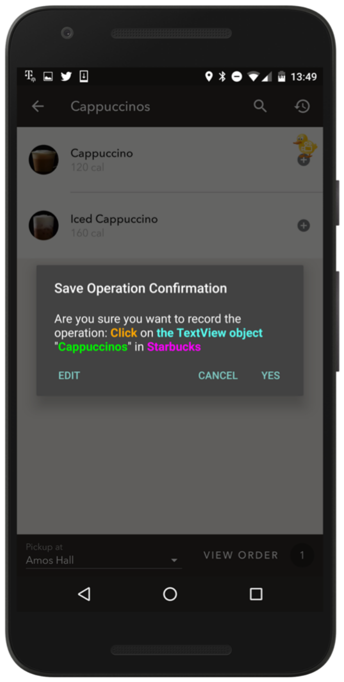
With SUGILITE, an intelligent agent can learn how to handle out-of-domain commands and perform the tasks using available third-party mobile apps in task-oriented dialogs from the user's demonstrations. More details about SUGILITE can be found in another paper [3]. In this position paper, we particularly focus on the design of its conversational interface.

The Conversational Interface of SUGILITE

In this section, we will briefly describe the conversational interface [1] of SUGILITE, and how SUGILITE uses the users' verbal instructions in generalizing the scripts. This generalization mechanism allows SUGILITE to learn how to perform tasks with different values for parameters (e.g., to order any Starbucks drink, or to find flights between any pair of cities) from a single demonstration. To better exhibit this interface, a video is also available[1].

To provide flexibility for users in different contexts, both creating the automation and running the automation can be performed through either the conversational interface or SUGILITE's own GUI. Sugilite currently uses Google ASR for voice recognition, and has its conversational interface based on LIA [1]. When the user gives a new voice command (e.g., "*order a cup of cappuccino*", "*check the score of the Steelers game*"), SUGILITE replies "*Sorry, but I don't understand… Would you like to teach me?*" (Figure 1).

After the user answers "*Yes*" in the conversational interface, SUGILITE replies "*When you say [COMMAND] … What shall I do? … Show me how to [COMMAND]*", switches to the home screen of the phone, and shows a popup to prompt the user to start demonstrating. As each step is demonstrated, SUGILITE pops up a confirmation (Figure 2). After the user finishes the demonstration, SUGILITE attempts to generalize the script by comparing the features of the target UI elements (in particular, their text labels) and any values typed into text fields against the words in the verbal command, trying to identify the user's parameters by matching the strings.



Save Operation Confirmation

Are you sure you want to record the operation: Click on the TextView object "Cappuccinos" in Starbucks

Figure 2: The recording confirmation popup of SUGILITE

For example, after the user demonstrates what to do for the command "*Order a pepperoni pizza*" using the Papa John's app, SUGILITE notices that the operation of clicking on the menu item "Pepperoni" from the "Papa's Picks" menu matches the word "pepperoni" in the verbal command (Figure 3). So SUGILITE can infer that "pepperoni" might be a parameter in this script, and "sausage" and "cheese pizza" are the two other possible values for this slot. As a result, SUGILITE learns how to properly handle the generalized command "Order a *[PIZZA_TYPE] pizza*" and can automatically order the right pizza based on the parameter given in the user's command.

In order to support this generalization, SUGILITE records the set of all possible alternatives to the UI element on which the user operates. SUGILITE finds these alternatives based on the UI structure, looking for other elements that are structurally in parallel with the selected element. This mechanism also allows SUGILITE to differentiate tasks with similar command structure but different values. For example, the commands "*Order Iced Cappuccino*" and "*Order a sausage pizza*" invoke different scripts, because the phrase "Iced Cappuccino" is among the alternative elements for an operation in one script, while "cheese pizza" is among the alternatives for a different script.

Planned Work

In the current version of SUGILITE, the verbal instruction from the user is only given *before* the demonstration. Then SUGILITE tries to generalize the script from the demonstration by using the user's verbal instruction as the intention of the user. Even though this simple approach works surprisingly well and can automatically generalize the scripts for many tasks with minimal additional user input [3], we want to go further and allow the users to give verbal instructions in conversations with the virtual agent and to perform demonstrations simultaneously while speaking. As future work, we wish to further explore how to better leverage (1) the verbal commands (and the narrations) given by the user; (2) the user's demonstrated actions, and (3) the screen contents shown on the user interface to understand the user's intentions and to learn the user's desired tasks [5].

In this workshop, we wish to discuss and gather feedback on designing the conversational user experience to address the following issues:

The Disambiguation / Data Description Problem
The data description problem is long-standing in PBD [2,4]. When the user demonstrates clicking on a screen item, it is difficult to determine what feature (e.g., text label, id, screen location, child elements, etc.) should be used for identifying which item to click on in future executions of the script. In a pilot study, we asked the users to narrate their actions while demonstrating. We found that the narrations are often very helpful in disambiguating the features (e.g., the users said things like "*click on the first item in the list*", "*click on the submit button*", "*choose the option with the lowest price*"). We plan to design a conversational user experience to allow the users to naturally narrate during the demonstration, and map the user narrations to each operation the users perform on the screen.

Procedure Editing
We would like to enable users to explicitly specify different parameter values for an existing script by speech. For example, the user should be able to give

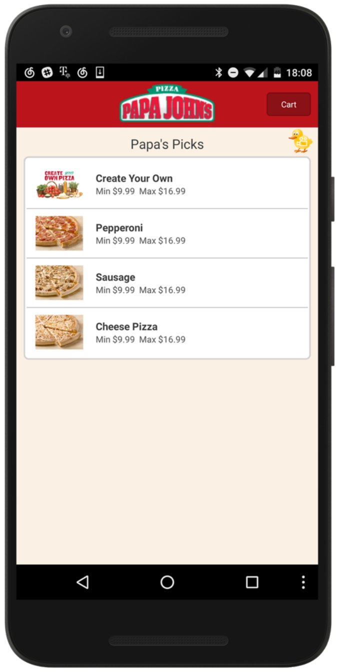


Figure 3: The Papa's Picks menu in the Papa John's app

Acknowledgement:
This work was supported by Yahoo! through CMU's InMind project.

commands like "*Get my regular breakfast order, except instead of iced coffee, I want hot coffee*." We wish to explore how the conversation in the mixed-initiative interface can be designed, leveraging the context of the existing scripts and GUI elements.

Conditionals and Triggers
In a preliminary study, participants from Amazon Mechanical Turk were asked to give example commands that they would like a smartphone intelligent personal assistant to perform. Many of their commands involved triggers based on the activity of the user. Some examples are: "*If I'm driving, send an auto-response for my incoming messages*", and "*Connect the wireless headphone if I'm running*". Many such commands also include conditionals, for example: "*order a hot cappuccino if it is cold outside, otherwise order an iced cappuccino*".

In some cases, the conditionals are implicit. For example, for "*what's the winning team for the Steeler's game*", the agent should return the name of the first team displayed on the interface if the score of the first team is higher than the second team, or the second team if the other way around.

We plan to investigate how the user would normally describe conditionals and triggers in the PBD scripts in a conversation through a further Mechanical Turk study. We will then explore how we can design the conversational interface to better support the end-user programming of conditionals and triggers.

Error Handling
SUGILITE often encounters two types of errors in its conversational interface: (1) errors in voice recognition and (2) recognizing the words correctly, but not being able to perform the task properly.

In light of the first type of error, we plan to enable SUGILITE to detect crucial actions (actions that cannot be undone), and ask users for confirmation before executing. This will ensure no critical action will be accidentally done in case of the misrecognition.

For the second type of error, currently when SUGILITE cannot find a matched item on the screen to operate on, it will pause and prompt the user to demonstrate the next action using direct manipulation. For the future, we plan to enable SUGILITE to ask questions like "*I don't see 'large' in the size option, I see grande, tall, etc., which one should I pick?*" in the conversation so the user can handle such errors by speech without necessarily having to touch the phone.

References

1. Amos Azaria, Jayant Krishnamurthy, and Tom M. Mitchell. 2016. Instructable intelligent personal agent. In *AAAI '16*
2. Allen Cypher and Daniel Conrad Halbert. 1993. *Watch what I do: programming by demonstration*. MIT press.
3. Toby Jia-Jun Li, Amos Azaria, and Brad A. Myers. 2017. SUGILITE: Creating Multimodal Smartphone Automation by Demonstration. In *CHI '17.* Retrieved from http://www.toby.li/sugilite_paper
4. Henry Lieberman. 2001. *Your wish is my command: Programming by example*. Morgan Kaufmann.
5. Ming Sun, Yun-Nung Chen, and Alexander I. Rudnicky. 2016. An Intelligent Assistant for High-Level Task Understanding. In *IUI '16*

Biography

Toby Jia-Jun Li (http://toby.li/) is a Ph.D. Student in the Human-Computer Interaction Institute at Carnegie Mellon University. His research interests include intelligent user interfaces, end-user programming, programming by demonstration and multi-modal interaction. His most recent research focuses on enabling end-users to teach intelligent agents new tasks using a combination of verbal instructions and demonstrations on user interfaces.

Brad A. Myers (http://www.cs.cmu.edu/~bam/) is a Professor in the Human-Computer Interaction Institute at Carnegie Mellon University. He will receive the CHI Lifetime Research Award this year, and is an IEEE Fellow, ACM Fellow, and member of the CHI Academy. His research interests include user interfaces, programming environments, programming by example, visual programming, and interaction techniques.

Amos Azaria (http://azariaa.com/) is a Senior Lecturer at Ariel University, Israel. His research interests include Human-agent interaction, Instructable Agents Machine Learning (Deep Learning), Natural Language Processing, Deception Detection, Modeling and Predicting Human behavior and Decision Making.

Igor Labutov is a postdoctoral associate at Carnegie Mellon University working with Tom Mitchell. His research focus is on building machine learning systems that can learn from rich human input in the form of natural language instruction and demonstration. Prior to starting at CMU, Igor received his PhD from Cornell University, where his focus was on developing machine learning algorithms for problems in education.

Alex Rudnicky (http://www.cs.cmu.edu/~air/) is a Research Professor at the Computer Science Department in the School of Computer Science at Carnegie Mellon University. His current interests center on language-based communication between humans and robots and on aspects of core speech recognition, such as out-of-vocabulary (OOV) word processing. He is also interested in approaches to learning based on implicit supervision and on improvement of speech system knowledge through dialog.

Tom M. Mitchell (http://www.cs.cmu.edu/~tom/) is a Professor at the Carnegie Mellon University. His research interests include Computer science, machine learning, artificial intelligence, and cognitive neuroscience. His research focuses on basic and applied problems in machine learning, on understanding how the human brain reads and represents the meaning of language, and statistical learning algorithms for natural language understanding by computer. He is a member of the United States National Academy of Engineering, a Fellow of the American Academy of Arts and Sciences, a Fellow of the American Association for the Advancement of Science and a Fellow of AAAI.