End User Programing of Intelligent Agents Using Demonstrations and Natural Language Instructions

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

ABSTRACT

End-user programmable intelligent agents that can learn new tasks and concepts from users' explicit instructions are desired. This paper presents our progress on expanding the capabilities of such agents in the areas of task applicability, task generalizability, user intent disambiguation and support for IoT devices through our multi-modal approach of combining programming by demonstration (PBD) with learning from natural language instructions. Our future directions include facilitating better script reuse and sharing, and supporting greater user expressiveness in instructions.

CCS CONCEPTS

• **Human-centered computing** → *Natural language interfaces*; *Interactive systems and tools.*

KEYWORDS

Programming by demonstration, end user development, multi-modal interaction, natural language programming.

ACM Reference Format:
Toby Jia-Jun Li. 2019. End User Programing of Intelligent Agents Using Demonstrations and Natural Language Instructions. In *24th International Conference on Intelligent User Interfaces (IUI '19 Companion), March 17–20, 2019, Marina del Rey, CA, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3308557.3308724

1 INTRODUCTION

Enabling end users to "teach" intelligent agents new tasks and concepts has become increasingly important due to the growing ubiquity of such agents residing in "smart" devices, such as phones, wearables, and speakers. Although these agents have a set of built-in functionalities, and most provide expandability by allowing users to add third-party "skills", they still fall short in the "long-tail" of tasks and suffer from the lack of customizability and flexibility [6].

From the user experience perspective, the lack of user programmability in agents results in frustration [8]. When a user gives an out-of-domain command, current agents either respond with a generic error message (e.g., '*'Sorry I don't understand.*") or perform a generic fallback action (e.g., a web search). Often, neither response is helpful – a more useful response would be to ask the user to instruct the agent how to perform the new task or to learn the new concept [8].

My research focuses on empowering end users to program agents for new tasks and concepts using a combination of programming by demonstration (PBD) and natural language instructions. Unlike learning systems that learn from passively collecting massive data from many users, my project focuses on interactive learning from a user's explicit demonstrations and instructions. While the input scale is small, carefully designed interactions can guide users to provide useful inputs for learning the generalized task procedure, the underlying user intentions, and the concepts involved.

The idea of using PBD in the end user development of task automation has been investigated in many contexts (e.g., [5]), but its adoption has been limited due to challenges of issues such generalization, reusability and applicability [2]. We are leveraging state-of-art natural language understanding techniques (e.g., [1]) to address those long-standing issues in PBD while preserving its low learning barrier. We used human-centered design methods [11] to achieve a balance between providing users with the necessary control, sufficient explainability and adequate expressive power instead of completely relying solely on AI techniques to perform the required generalizations and inferences.

2 RESEARCH PROGRESS UP TO DATE

2.1 Task Applicability and Generalizability

As a major foundation of my Ph.D. research, Sugilite [6] addresses two limitations in prior PBD work: applicability, and generalizability. For applicability, Sugilite supports users in programming tasks across diverse domains by allowing automating procedures and extracting information from one or multiple third-party Android apps. It can extract a UI snapshot knowledge graph [7] from every screen in each app, from which it infers a generalized query used for manipulating and extracting information from the app's GUI through the Android Accessibility API.

Sugilite's novel multi-modal interface leverages the user's natural language description of the task and the GUI structure of apps to support automatic generalization of recorded procedures through parameterization. For example, if the user describes the task as "order a cup of cappuccino" and demonstrates its procedure in the Starbucks app, Sugilite will identify "cappuccino" as a parameter by observing the user choosing it from a list of available drinks in the Starbucks app GUI. Sugilite extracts other possible values for this parameter from different branches in the GUI so that the user can perform, for example, "order a cup of caramel macchiato" with Sugilite without having to teach it again, even when "caramel macchiato" is in a different subsection of the menu from "cappuccino".

IUI '19 Companion, March 17–20, 2019, Marina del Rey, CA, USA

ACM ISBN 978-1-4503-6673-1/19/03…$15.00
https://doi.org/10.1145/3308557.3308724

In our lab evaluation, SUGILITE was shown to be usable and useful even for users with no programming background.

2.2 Natural Language for Data Description Disambiguation

Another long-standing challenge in PBD is to deal with ambiguities in demonstrations. From the demonstrations, the agent should produce more than a literal record-and-replay macro, but instead learn the task at a higher level of abstraction so it can perform similar tasks in new contexts. For example, suppose the user demonstrates choosing a restaurant from a search results list, the chosen item can have many properties. (e.g., screen location, relative position in the list, or review rating) The agent needs to understand the user's intention to correctly learn what to select in the context of the task. This is also known as the *data description problem* [2].

Our APPINITE [7] interface addresses this challenge using *mutual disambiguation* [12] in multi-modal interaction, where APPINITE asks the user to provide a natural language explanation (e.g., "*choose the highest rated restaurant within a mile*" for the previous example) for an ambiguous demonstration. APPINITE uses the explanation to disambiguate the demonstration to create a data description that reflects the user's underlying intention. Meanwhile, the demonstration and the GUI are also used for grounding and disambiguating the verbal explanation for semantic parsing. If the explanation is still ambiguous, APPINITE engages in multi-turn mixed-initiative dialogs with the user to resolve the data description, and provides interactive visualization over the app's GUIs to help the user focus on explaining the key aspects that are useful for the disambiguation.

Compared with prior systems with AI-based programming synthesis methods (e.g., [3, 10]) running on multiple examples, APPINITE's approach for resolving data descriptions provides higher transparency and explainability, more user control, and greater user expressiveness, as well as better usability by not requiring users to provide multiple *meaningfully* different examples for inferring the programming logic, which has been shown to be difficult and error-prone for non-programmers [4]. Our study has shown that APPINITE's approach is feasible for end users.

2.3 Support for Smart Home and IoT Devices

In the EPIDOSITE extension [9] to our SUGILITE agent, we expanded the domain to support tasks for smart home devices. The main motivation is interoperability – current agents have limited support for tasks involving multiple devices as sensors and actuators, because they only support devices from the same company, within the same "eco-system", or providing open-access APIs. To address this, EPIDOSITE uses the smartphone as a hub for devices, controlling and reading data from them through the corresponding mobile apps.

3 FUTURE DIRECTIONS

3.1 Reusability and Shareability

First, we seek to enable the agent to learn reusable and transferable sub-concepts and procedures through a new top-down instruction framework, where the agent can break the task into smaller pieces and learn them independently, where each component may be reused individually in different tasks. Second, we will design new representations for learned procedures and new interfaces for editing existing procedures to handle different situations. Third, we also plan to enable secure and privacy-preserving knowledge sharing, where the agent can remove private information, and generalize learned procedures and knowledge so that they can be used by other users.

3.2 Script Expressiveness

We are also working to raise the ceiling of our agent, so it can learn more complex tasks. We are especially interested in enabling users to instruct the agent about flexible control structures such as conditionals, loops, triggers and exceptions.

We have conducted a study on how users naturally express these structures verbally in the context of mobile apps [13]. Based on the findings, we are designing new interfaces to provide users with greater expressiveness in instructions by enabling them to specify more sophisticated control structures.

3.3 Deployment Study

Based on our lab studies, the new features in our agent have been shown to be usable and useful. We look forward to conducting a field study to understand how users use our system in real contexts. Our current prototype has already been open-sourced, and deployed in small-scale through the Google Play Store. We eventually plan to release our software to the public for general use.

ACKNOWLEDGMENTS

I would like to thank my Ph.D. advisor Prof. Brad Myers for his support and guidance. This research was supported in part by Oath through the InMind project and in part by NSF grant IIS-1814472.

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] Sumit Gulwani. 2011. Automating String Processing in Spreadsheets Using Input-output Examples. In *POPL '11*.
[4] Tak Yeon Lee, Casey Dugan, and Benjamin B. Bederson. 2017. Towards Understanding Human Mistakes of Programming by Example: An Online User Study. In *IUI '17*.
[5] Gilly Leshed, Eben M. Haber, Tara Matthews, and Tessa Lau. 2008. CoScripter: Automating & Sharing How-to Knowledge in the Enterprise. In *CHI '08*.
[6] Toby Jia-Jun Li, Amos Azaria, and Brad A. Myers. 2017. SUGILITE: Creating Multimodal Smartphone Automation by Demonstration. In *CHI '17*.
[7] Toby Jia-Jun Li, Igor Labutov, Xiaohan Nancy Li, Xiaoyi Zhang, Wenze Shi, Tom M. Mitchell, and Brad A. Myers. 2018. APPINITE: A Multi-Modal Interface for Specifying Data Descriptions in Programming by Demonstration Using Verbal Instructions. In *VL/HCC '18*.
[8] Toby Jia-Jun Li, Igor Labutov, Brad A. Myers, Amos Azaria, Alexander I. Rudnicky, and Tom M. Mitchell. 2018. Teaching Agents When They Fail: End User Development in Goal-oriented Conversational Agents. In *Studies in Conversational UX Design*. Springer.
[9] Toby Jia-Jun Li, Yuanchun Li, Fanglin Chen, and Brad A. Myers. 2017. Programming IoT Devices by Demonstration Using Mobile Apps. In *IS-EUD '17*.
[10] Toby Jia-Jun Li and Oriana Riva. 2018. KITE: Building conversational bots from mobile apps. In *MobiSys '18*.
[11] Brad A. Myers, Andrew J. Ko, Chris Scaffidi, Stephen Oney, YoungSeok Yoon, Kerry Chang, Mary Beth Kery, and Toby Jia-Jun Li. 2017. Making End User Development More Natural. In *New Perspectives in End-User Development*.
[12] Sharon Oviatt. 1999. Mutual disambiguation of recognition errors in a multimodel architecture. In *CHI '99*.
[13] Marissa Radensky, Toby Jia-Jun Li, and Brad A. Myers. 2018. How End Users Express Conditionals in Programming by Demonstration for Mobile Apps. In *VL/HCC '18*.