Interactive Task Learning from GUI-Grounded Natural Language Instructions and Demonstrations

Toby Jia-Jun Li, Tom M. Mitchell, Brad A. Myers
School of Computer Science
Carnegie Mellon University
{tobyli, tom.mitchell, bam}@cs.cmu.edu

Abstract

We show SUGILITE, an intelligent task automation agent that can learn new tasks and relevant associated concepts interactively from the user's natural language instructions and demonstrations, using the graphical user interfaces (GUIs) of third-party mobile apps. This system provides several interesting features: (1) it allows users to teach new task procedures and concepts through verbal instructions together with demonstration of the steps of a script using GUIs; (2) it supports users in clarifying their intents for demonstrated actions using GUI-grounded verbal instructions; (3) it infers parameters of tasks and their possible values in utterances using the hierarchical structures of the underlying app GUIs; and (4) it generalizes taught concepts to different contexts and task domains. We describe the architecture of the SUGILITE system, explain the design and implementation of its key features, and show a prototype in the form of a conversational assistant on Android.

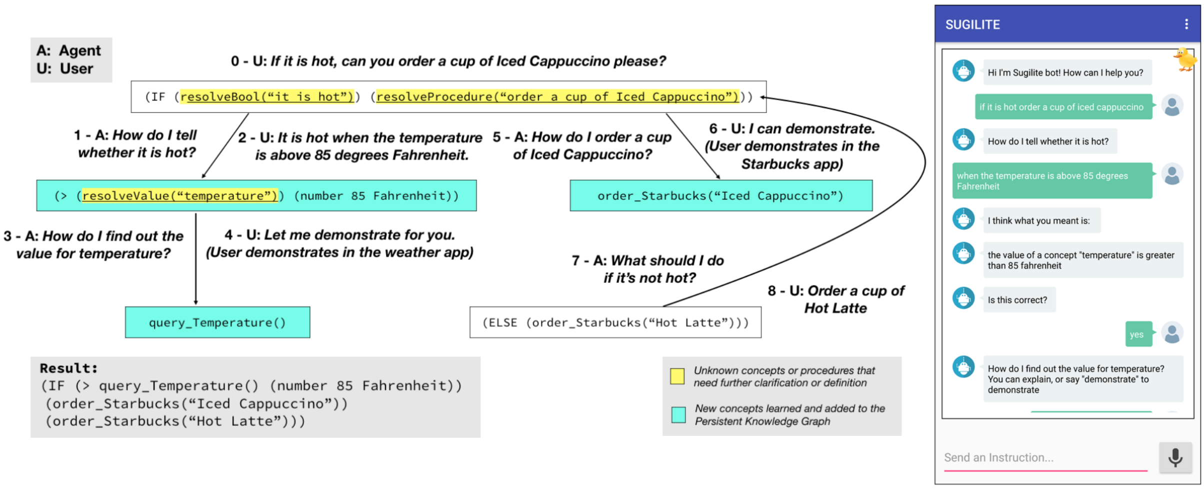
1 Introduction

Interactive task learning (ITL) is an emerging research topic that focuses on enabling task automation agents to learn new tasks and their corresponding relevant concepts through natural interaction with human users (Laird et al., 2017). This topic is also known as *end user development* (EUD) for task automation (Ko et al., 2011; Myers et al., 2017). Work in this domain includes both physical agents (e.g., robots) that learn tasks that might involve sensing and manipulating objects in the real world (Chai et al., 2018; Argall et al., 2009), as well as software agents that learn how to perform tasks through software interfaces (Azaria et al., 2016; Allen et al., 2007; Labutov et al., 2018; Leshed et al., 2008). This paper focuses on the latter category.

A particularly useful application of ITL is for conversational virtual assistants (e.g., Apple Siri, Google Assistant, Amazon Alexa). These systems have been widely adopted by end users to perform tasks in a variety of domains through natural language conversation. However, a key limitation of these systems is that their task fulfillment and language understanding capabilities are limited to a small set of pre-programmed tasks (Li et al., 2018b; Labutov et al., 2018). This limited support is not adequate for the diverse "long-tail" of user needs and preferences (Li et al., 2017a). Although some software agents provide APIs to enable third-party developers to develop new "skills" for them, this requires significant programming expertise and relevant APIs, and therefore is not usable by the vast majority of end users.

Natural language instructions play a key role in some ITL systems for virtual assistants, because this modality represents an natural way for humans to teach new tasks (often to other humans), and has a low learning barrier compared to existing textual or visual programming languages for task automation. Some prior systems (Azaria et al., 2016; Labutov et al., 2018; Le et al., 2013; Srivastava et al., 2017, 2018) relied solely natural language instruction, while others (Allen et al., 2007; Kirk and Laird, 2019; Sereshkeh et al., 2020) also used demonstrations of direct manipulations to supplement the natural language instructions. We surveyed the prior work, and identified the following five key design challenges:

1. **Usability**: The system should be usable for users without significant programming expertise. It should be easy and intuitive to use with a low learning barrier. This requires careful design of the dialog flow to best match the user's natural model of task instruction.
2. **Applicability**: The system should handle a


A: Agent
U: User
0 - U: If it is hot, can you order a cup of Iced Cappuccino please?
(IF (resolveBool("it is hot")) (resolveProcedure("order a cup of Iced Cappuccino")))
1 - A: How do I tell whether it is hot?
2 - U: It is hot when the temperature is above 85 degrees Fahrenheit.
(> (resolveValue("temperature")) (number 85 Fahrenheit))
3 - A: How do I find out the value for temperature?
4 - U: Let me demonstrate for you. (User demonstrates in the weather app)
query_Temperature()
5 - A: How do I order a cup of Iced Cappuccino?
6 - U: I can demonstrate. (User demonstrates in the Starbucks app)
order_Starbucks("Iced Cappuccino")
7 - A: What should I do if it's not hot?
8 - U: Order a cup of Hot Latte
(ELSE (order_Starbucks("Hot Latte")))
Unknown concepts or procedures that need further clarification or definition
New concepts learned and added to the Persistent Knowledge Graph
SUGILITE
Send an Instruction...

```
Result:
(IF (> query_Temperature() (number 85 Fahrenheit))
 (order_Starbucks("Iced Cappuccino"))
 (order_Starbucks("Hot Latte")))
```

Hi I'm Sugilite bot! How can I help you?

if it is hot order a cup of iced cappuccino

How do I tell whether it is hot?

when the temperature is above 85 degrees Fahrenheit

I think what you meant is:

the value of a concept "temperature" is greater than 85 fahrenheit

Is this correct?

yes

How do I find out the value for temperature? You can explain, or say "demonstrate" to demonstrate

Figure 1: An example dialog structure while SUGILITE learns a new task that contains a conditional and new concepts. The numbers indicate the sequence of the utterances. The screenshot on the right shows the conversational interface during these steps.

wide range of common and long-tail tasks across different domains. Many existing systems can only work with pre-specified task domains (Labutov et al., 2018; Azaria et al., 2016; Gulwani and Marron, 2014), or services that provide open API access to their functionalities (Campagna et al., 2017; Le et al., 2013). This limits the applicability of those systems to a smaller subset of tasks.

The same problem also applies to the language understanding capability of the system. It should be able to understand, ground, and act upon instructions in different task domains (e.g., different phone apps) without requiring pre-built parsers for each domain.

3. **Generalizability**: The system should learn generalized procedures and concepts to handle new task contexts that go beyond the example context used for instruction. This includes inferring parameters of tasks, allowing the use of different parameter values, and adapting learned concepts to new task domains.

4. **Flexibility**: The system should be sufficiently expressive to allow users to specify flexible rules, conditions, and other control structures that reflect their intentions.

5. **Robustness**: The system should be resilient to minor changes in target applications, and be able to recover from errors caused by previously unseen or unexpected situations, possibly with some help from the user.

To address these challenges, we present the prototype of a new task automation agent named SUGILITE[1][2]. This prototype integrates and implements the results from several of our prior research works (Li et al., 2017a, 2018a, 2017b; Li and Riva, 2018; Li et al., 2019), and we are current preparing for a field deployment study with this prototype. The implementation of our system is also open-sourced on GitHub[3]. The high-level approach used in SUGILITE is to combine conversational natural language instructions with demonstrations on mobile app GUIs, and to use each of the two modalities to disambiguate, ground, and supplement the user's inputs from the other modality through mixed-initiative interactions.

2 System Overview

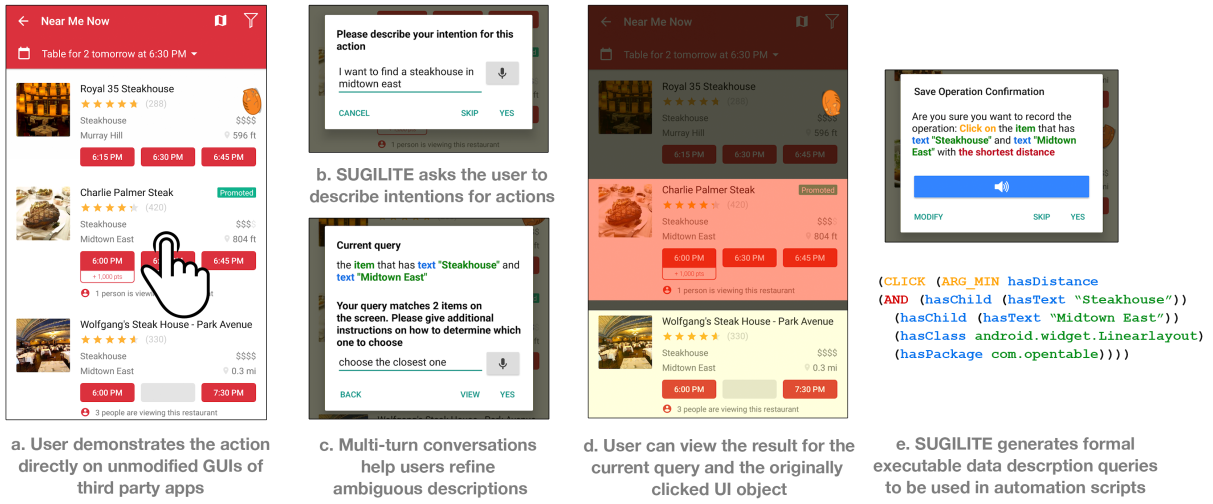
This section explains how SUGILITE learns new tasks and concepts from the multi-modal interactive instructions from the users.

The user starts with speaking a command. The command can describe either an action (e.g., "check the weather") or an automation rule with a condition (e.g., "If it is hot, order a cup of Iced Cappuccino"). Suppose that the agent has no prior knowledge in any of the involved task

[1] Sugilite is a gemstone, and here stands for **S**martphone **U**sers **G**enerating **I**ntelligent **L**ikeable **I**nterfaces **T**hrough **E**xamples.

[2] A demo video is available at https://www.youtube.com/watch?v=tdHEk-GeaqE

[3] https://github.com/tobyli/Sugilite_development



a. User demonstrates the action directly on unmodified GUIs of third party apps

Please describe your intention for this action

I want to find a steakhouse in midtown east

b. SUGILITE asks the user to describe intentions for actions

Current query

the item that has text "Steakhouse" and text "Midtown East"

Your query matches 2 items on the screen. Please give additional instructions on how to determine which one to choose

choose the closest one

c. Multi-turn conversations help users refine ambiguous descriptions

d. User can view the result for the current query and the originally clicked UI object

Save Operation Confirmation

Are you sure you want to record the operation: Click on the item that has text "Steakhouse" and text "Midtown East" with the shortest distance

```
(CLICK (ARG_MIN hasDistance
(AND (hasChild (hasText “Steakhouse”))
  (hasChild (hasText “Midtown East”))
  (hasClass android.widget.Linearlayout)
  (hasPackage com.opentable))))
```

e. SUGILITE generates formal executable data descrption queries to be used in automation scripts

Figure 2: The screenshots of SUGILITE's demonstration mechanism and its multi-modal mixed-initiative intent clarification process for the demonstrated actions.

domains, then it will recursively resolve the unknown concepts and procedures used in the command. Although it does not know these concepts, it can recognize the structure of the command (e.g., conditional), and parse each part of the command into the corresponding typed `resolve` functions, as shown in Figure 1. SUGILITE uses a grammar-based executable semantic parsing architecture (Liang, 2016); therefore its conversation flow operates on the recursive execution of the `resolve` functions. Since the `resolve` functions are typed, the agent can generate prompts based on their types (e.g., “How do I tell whether…” for `resolveBool` and “How do I find out the value for…” for `resolveValue`).

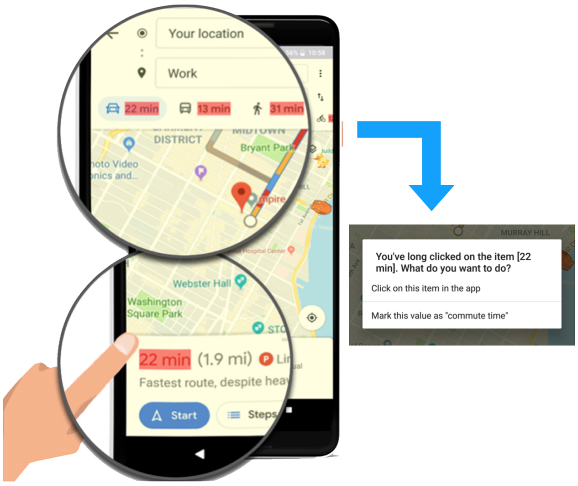
When the SUGILITE agent reaches the `resolve` function for a value query or a procedure, it asks the users if they can demonstrate them. The users can then demonstrate how they would normally look up the value, or perform the procedure manually with existing mobile apps on the phone by direct manipulation (Figure 2a). For any ambiguous demonstrated action, the user verbally explains the intent behind the action through multi-turn conversations with the help from an interaction proxy overlay that guides the user to focus on providing more effective input (see Figure 2bcde, more details in Section 3.2). When the user demonstrates a value query (e.g., finding out the value of the temperature), SUGILITE highlights the GUI elements showing values with the compatible types (see Figure 3) to assist the user in finding the appropriate GUI element during the demonstration.

All user-instructed value concepts, Boolean concepts, and procedures automatically get generalized by SUGILITE. The procedures are parameterized so that they can be reused with different parameter values in the future. For example, for Utterance 8 in Figure 1, the user does not need to demonstrate again since the system can invoke the newly-learned `order_Starbucks` function with a different parameter value (details in Section 3.3). The learned concepts and value queries are also generalized so that the system recognizes the different definitions of concepts like “hot” and value queries like “temperature” in different contexts (details in Section 3.4).

3 Key Features

3.1 Using Demonstrations in Natural Language Instructions

SUGILITE allows users to use demonstrations to teach the agent any unknown procedures and concepts in their natural language instructions. As discussed earlier, a major challenge in ITL is that understanding natural language instructions and carrying out the tasks accordingly require having knowledge in the specific task domains. Our use of programming by demonstration (PBD) is an effective way to address this “out-of-domain” problem in both the task-fulfillment and the natural language understanding processes (Li et al., 2018b). In SUGILITE, procedural actions are represented as sequences of GUI operations, and declarative con-



You've long clicked on the item [22 min]. What do you want to do?

Click on this item in the app

Mark this value as "commute time"

Figure 3: The user teaches the value concept "commute time" by demonstrating querying the value in Google Maps. SUGILITE highlights all the duration values on the Google Maps GUI.

cepts can be represented as references to GUI contents. This approach supports ITL for a wide range of tasks – virtually anything that can be performed with one or more existing third-party mobile apps.

Our prior study (Li et al., 2019) also found that the availability of app GUI references can result in end users providing clearer natural language commands. In one study where we asked participants to instruct an intelligent agent to complete everyday computing tasks in natural language, the participants who saw screenshots of relevant apps used *fewer* unclear, vague, or ambiguous concepts in their verbal instructions than those who did not see the screenshots. Details of the study design and the results can be found in Li et al. (2019).

3.2 Spoken Intent Clarification for Demonstrated Actions

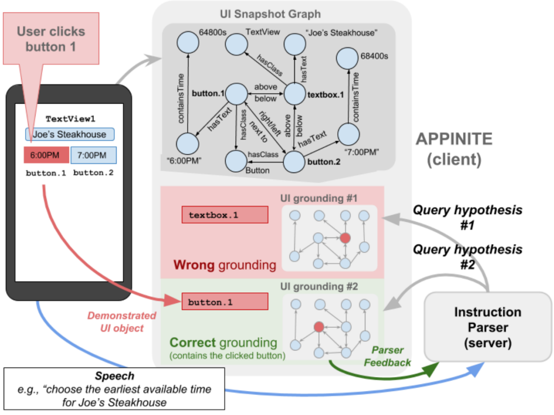
A major limitation of demonstrations is that they are too literal, and are therefore brittle to any changes in the task context. They encapsulate *what* the user did, but not *why* the user did it. When the context changes, the agent often may not know what to do, due to this lack of understanding of the user intents behind their demonstrated actions. This is known as the *data description problem* in the PBD community, and it is regarded as a key problem in PBD research (Cypher and Halbert, 1993; Lieberman, 2001). For example, just looking at the action shown in Figure 2a, one cannot tell if the user meant "the restaurant with the most reviews", "the promoted restaurant", "the restaurant with 1,000 bonus points", "the cheapest Steakhouse", or any other criteria, so the system cannot generate a description for this action that accurately reflects the user's intent. A prior approach is to ask for multiple examples from the users (McDaniel and Myers, 1999), but this is often not feasible due to the user's inability to come up with useful and complete examples, and the amount of examples required for complex tasks (Myers and McDaniel, 2001; Lee et al., 2017).

SUGILITE's approach is to ask users to verbally explain their intent for the demonstrated actions using speech. Our formative study (Li et al., 2018a) found that end users were able to provide useful and generalizable explanations for the intents of their demonstrated actions. They also commonly used in their utterances semantic references to GUI contents (e.g., "the close by restaurant" for an entry showing the text "596 ft") and implicit spatial references (e.g., "the score for Lakers" for a text object that contains a numeric value and is right-aligned to another text object "Lakers").

Based on these findings, we designed and implemented a multi-modal mixed-initiative intent clarification mechanism for demonstrated actions. As shown in Figure 2, the user describes their intention in natural language, and iteratively refines the descriptions to remove ambiguity with the help of an interactive overlay (Figure 2d). The overlay highlights the result from executing the current data description query, and helps the user focus on explaining the key differences between the target object (highlighted in red) and the false positives (highlighted in yellow) of the query.

To ground the user's natural language explanations about GUI elements, SUGILITE represents each GUI screen as a *UI snapshot graph*. This graph captures the GUI elements' text labels, meta-information (including screen position, type, and package name), and the spatial (e.g., `nextTo`), hierarchical (e.g., `hasChild`), and semantic relations (e.g., `containsPrice`) among them (Figure 4). A semantic parser translates the user's explanation into a graph query on the UI snapshot graph, executes it on the graph, and verifies if the result matches the correct entity that the user originally demonstrated. The goal of this process is to generate a query that uniquely matches the target UI element and also reflects the user's underlying intent.

Our semantic parser uses a Floating Parser ar-



```
(CLICK (ARG_MIN numeric_time (AND
  (hasClass Button)
  (below (hasText “Joe’s Steakhouse”)))))
```

Figure 4: SUGILITE's instruction parsing and grounding process for intent clarifications illustrated on an example UI snapshot graph constructed from a simplified GUI snippet.

chitecture (Pasupat and Liang, 2015) and is implemented with the SEMPRE framework (Berant et al., 2013). We represent UI snapshot graph queries in a simple but flexible LISP-like query language (S-expressions) that can represent joins, conjunctions, superlatives and their compositions, constructed by the following 7 grammar rules:

$$E \to e; E \to S; S \to (\texttt{join}\ r\ E); S \to (\texttt{and}\ S\ S)$$

$$T \to (\texttt{ARG\_MAX}\ r\ S); T \to (\texttt{ARG\_MIN}\ r\ S); Q \to S \mid T$$

where Q is the root non-terminal of the query expression, e is a terminal that represents a UI object entity, r is a terminal that represents a relation, and the rest of the non-terminals are used for intermediate derivations. SUGILITE's language forms a subset of a more general formalism known as Lambda Dependency-based Compositional Semantics (Liang et al., 2013), which is a notationally simpler alternative to lambda calculus which is particularly well-suited for expressing queries over knowledge graphs. More technical details and the user evaluation are discussed in Li et al. (2018a).

3.3 Task Parameterization through GUI Grounding

Another way SUGILITE leverages GUI groundings in the natural language instructions is to infer task parameters and their possible values. This allows the agent to learn generalized procedures (e.g., to order *any kind of beverage* from Starbucks) from a demonstration of a specific instance of the task (e.g., ordering an iced cappuccino).

SUGILITE achieves this by comparing the user utterance (e.g., "order a cup of iced cappuccino") against the *data descriptions* of the target UI elements (e.g., click on the menu item that has the text "Iced Cappuccino") and the arguments (e.g., put "Iced Cappuccino" into a search box) of the demonstrated actions for matches. This process grounds different parts in the utterances to specific actions in the demonstrated procedure. It then analyzes the hierarchical structure of GUI at the time of demonstration, and looks for alternative GUI elements that are in parallel to the original target GUI element structurally. In this way, it extracts the other possible values for the identified parameter, such as the names of all the other drinks displayed in the same menu as "Iced Cappuccino"

The extracted sets of possible parameter values are also used for disambiguating the procedures to invoke, such as invoking the `order_Starbucks` procedure for the command "order a cup of *latte*", but invoking the `order_PapaJohns` procedure for the command "order a *cheese pizza*."

3.4 Generalizing the Learned Concepts

In addition to the procedures, SUGILITE also automatically generalizes the learned *concepts* in order to reuse parts of existing concepts as much as possible to avoid requiring users to perform redundant demonstrations (Li et al., 2019).

For Boolean concepts, SUGILITE assumes that the Boolean operation and the types of the arguments stay the same, but the arguments themselves may differ. For example, the concept "hot" used in Figure 1 can be generalize to "`arg0` is greater than `arg1`" where `arg0` and `arg1` can be value queries or constant values of the *temperature type*. This allows the various constant thresholds of temperature, or dynamic queries for temperatures depending on the specific task context. This mechanism allows concepts to be used across different contexts (e.g., determining whether to order iced coffee vs. whether to open the window) task domains (e.g., "the weather is hot" vs. "the oven is hot").

Similarly, named value queries (resolved from `resolveValue` such as "temperature" in Figure 1) can be generalized to have different implementations depending on the task domain. In "the temperature outside", `query_Temperature()`

can invoke the weather app, whereas in "the temperature of the oven" it can invoke the smart oven app to look up the current temperature of the oven (Li et al., 2017b).

4 Evaluation

We conducted several lab user studies to evaluate the usability, efficiency and effectiveness of SUGILITE. The results of these study showed that end users without significant programming expertise were able to successfully teach the agent the procedures of performing common tasks (e.g., ordering pizza, requesting Uber, checking sports score, ordering coffee) (Li et al., 2017a), conditional rules for triggering the tasks (Li et al., 2019), and concepts relevant to the tasks (e.g., the weather is *hot*, the traffic is *heavy*) (Li et al., 2019) using SUGILITE. The users were also able to clarify their intents when ambiguities arise (Li et al., 2018a). Most of our participants found SUGILITE easy and natural to use (Li et al., 2017a, 2018a, 2019). Efficiency wise, teaching a task usually took the user 3–6 times longer than how long it took to perform the task manually in our studies (Li et al., 2017a), which indicates that teaching a task using SUGILITE can save time for many repetitive tasks.

5 Discussion and Future Work

5.1 Using GUIs for Language Grounding

SUGILITE illustrates the great promise of using GUIs as a resource for grounding and understanding natural language instructions in ITL. The GUIs encapsulate rich knowledge about the flows of the underlying tasks and the properties and relations of relevant entities, so they can be used to bootstrap the domain-specific knowledge needed by ITL systems that rely on natural language instructions for learning. Users are also familiar with GUIs, which makes GUIs the ideal medium to which users can refer during task instructions. A major challenge in natural language instruction is that the users do not know what concepts or knowledge the agent already knows so that they can use it in their instructions (Li et al., 2019). Therefore, they often introduce additional unknown concepts that are either unnecessary or entirely beyond the capability of the agent (e.g., explaining "hot" as "when I'm sweating" when teaching the agent to "open the window when it is hot"). By using the app GUIs as the medium, the system can effectively constrain the users to refer to things that can be found out from some app GUIs (e.g., "hot" can mean "the temperature is high"), which mostly overlaps with the "capability ceiling" of smartphone-based agents, and allows the users to define new concepts for the agent by referring to app GUIs (Li et al., 2017a, 2019).

5.2 More Robust Natural Language Understanding

The current version of SUGILITE uses a grammar-based executable semantic parser to understand the users' natural language explanations of their intents for the demonstrated actions. While this approach comes with many benefits, such as only requiring a small amount of training data and not relying on any domain knowledge, it has rigid patterns and therefore sometimes encounters problems with the flexible structures and varied expressions in the user utterances.

We are looking at alternative approaches for parsing natural language instructions into our domain-specific language (DSL) for representing data description queries and task execution procedures. A promising strategy is to take advantage of the abstract syntax tree (AST) structure in our DSL for constructing a neural parser (Xu et al., 2020; Yin and Neubig, 2017), which reduces the amount of training data needed and enforces the well-formedness of the output code.

The current model also only uses the semantic information from the local user instructions and their corresponding app GUIs. Another promising approach to enable more robust natural language understanding is to leverage the pre-trained general-purpose language models (e.g., BERT (Devlin et al., 2018)) to encode the user instructions and the information extracted from app GUIs.

5.3 Extracting Task Semantics from GUIs

An interesting future direction is to better extract semantics from app GUIs so that the user can focus on high-level task specifications and personal preferences without dealing with low-level mundane details (e.g., "buy 2 burgers" means setting the value of the textbox below the text "quantity" and next to the text "Burger" to be "2"). Some works have made early progress in this domain (Liu et al., 2018b; Deka et al., 2016; Chen et al., 2020) thanks to the availability of large datasets of GUIs like RICO (Deka et al., 2017). Recent reinforcement learning-based approaches and semantic parsing techniques have also shown promising results in learning models for navigating through GUIs for user-specified task objectives (Liu et al., 2018a; Pasupat et al., 2018). For ITL, an interesting future challenge is to combine these user-independent domain-agnostic machine-learned models with the user's personalized instructions for a specific task. This will likely require a new kind of mixed-initiative instruction (Horvitz, 1999) where the agent is more proactive in guiding the user and takes more initiative in the dialog. This could be supported by improved background knowledge and task models, and more flexible dialog frameworks that can handle the continuous refinement and uncertainty inherent in natural language interaction, and the variations in user goals. Collecting and aggregating personal task instructions across many users also introduce important concerns on user privacy, as discussed in (Li et al., 2020).

5.4 Multi-Modal Interactions in Conversational Learning

SUGILITE combines speech and direct manipulation to enable a "speak and point" interaction style, which has been studied since early interactive systems like Put-That-There (Bolt, 1980). As described in Section 3.2, a key pattern used in SUGILITE's multi-modal interface is *mutual disambiguation* (Oviatt, 1999) where it utilizes inputs in complementary modalities to infer robust and generalizable scripts that can accurately represent user intentions.

We are currently exploring other ways of using multi-modal interactions to supplement natural language instructions in ITL. A promising direction is to use GUI references to help with repairing conversational breakdowns (Beneteau et al., 2019; Ashktorab et al., 2019; Myers et al., 2018) caused by incorrect semantic parsing, intent classification, or entity recognition. Since GUIs encapsulate rich semantic information about the users' intents, the task flows, and the task constraints, we can potentially ask the users to point to the relevant GUI screens as a part of the error handling process, explaining the errors with references to the GUIs, and helping the system recover from the breakdowns.

6 Conclusion

We described SUGILITE, a task automation agent that can learn new tasks and relevant concepts interactively from users through their GUI-grounded natural language instructions and demonstrations. This system provides capabilities such as intent clarification, task parameterization, and concept generalization. SUGILITE shows the promise of using app GUIs for grounding natural language instructions, and the effectiveness of resolving unknown concepts, ambiguities, and vagueness in natural language instructions using a mixed-initiative multi-modal approach.

Acknowledgments

This research was supported in part by Verizon and Oath through the InMind project, a J.P. Morgan Faculty Research Award, NSF grant IIS-1814472, and AFOSR grant FA95501710218. Any opinions, findings or recommendations expressed here are those of the authors and do not necessarily reflect views of the sponsors. We thank Amos Azaria, Igor Labutov, Xiaohan Nancy Li, Xiaoyi Zhang, Jingya Chen, and Marissa Radensky for their contributions to the development of this system.

References

James Allen, Nathanael Chambers, George Ferguson, Lucian Galescu, Hyuckchul Jung, Mary Swift, and William Taysom. 2007. PLOW: A Collaborative Task Learning Agent. In *Proceedings of the 22nd National Conference on Artificial Intelligence - Volume 2*, AAAI'07, pages 1514–1519, Vancouver, British Columbia, Canada. AAAI Press.

Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. 2009. A Survey of Robot Learning from Demonstration. *Robot. Auton. Syst.*, 57(5):469–483.

Zahra Ashktorab, Mohit Jain, Q Vera Liao, and Justin D Weisz. 2019. Resilient chatbots: Repair strategy preferences for conversational breakdowns. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, page 254. ACM.

Amos Azaria, Jayant Krishnamurthy, and Tom M. Mitchell. 2016. Instructable Intelligent Personal Agent. In *Proc. The 30th AAAI Conference on Artificial Intelligence (AAAI)*, volume 4.

Erin Beneteau, Olivia K Richards, Mingrui Zhang, Julie A Kientz, Jason Yip, and Alexis Hiniker. 2019. Communication breakdowns between families and alexa. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–13.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544.

Richard A. Bolt. 1980. “Put-that-there”: Voice and Gesture at the Graphics Interface. In *Proceedings of the 7th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH ’80, pages 262–270, New York, NY, USA. ACM.

Giovanni Campagna, Rakesh Ramesh, Silei Xu, Michael Fischer, and Monica S. Lam. 2017. Almond: The architecture of an open, crowdsourced, privacy-preserving, programmable virtual assistant. In *Proceedings of the 26th International Conference on World Wide Web*, pages 341–350.

Joyce Y Chai, Qiaozi Gao, Lanbo She, Shaohua Yang, Sari Saba-Sadiya, and Guangyue Xu. 2018. Language to action: Towards interactive task learning with physical agents. In *IJCAI*, pages 2–9.

Jieshan Chen, Chunyang Chen, Zhenchang Xing, Xiwei Xu, Liming Zhu, Guoqiang Li, and Jinshui Wang. 2020. Unblind your apps: Predicting natural-language labels for mobile gui components by deep learning. In *Proceedings of the 42nd International Conference on Software Engineering*, ICSE ’20.

Allen Cypher and Daniel Conrad Halbert. 1993. *Watch what I do: programming by demonstration*. MIT press.

Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hibschman, Daniel Afergan, Yang Li, Jeffrey Nichols, and Ranjitha Kumar. 2017. Rico: A Mobile App Dataset for Building Data-Driven Design Applications. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, UIST ’17, pages 845–854, New York, NY, USA. ACM.

Biplab Deka, Zifeng Huang, and Ranjitha Kumar. 2016. ERICA: Interaction Mining Mobile Apps. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, UIST ’16, pages 767–776, New York, NY, USA. ACM.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Sumit Gulwani and Mark Marron. 2014. Nlyze: Interactive programming by natural language for spreadsheet data analysis and manipulation. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 803–814.

Eric Horvitz. 1999. Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 159–166. ACM.

James R. Kirk and John E. Laird. 2019. Learning hierarchical symbolic representations to support interactive task learning and knowledge transfer. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 6095–6102. International Joint Conferences on Artificial Intelligence Organization.

Amy J. Ko, Robin Abraham, Laura Beckwith, Alan Blackwell, Margaret Burnett, Martin Erwig, Chris Scaffidi, Joseph Lawrance, Henry Lieberman, Brad Myers, Mary Beth Rosson, Gregg Rothermel, Mary Shaw, and Susan Wiedenbeck. 2011. The State of the Art in End-user Software Engineering. *ACM Comput. Surv.*, 43(3):21:1–21:44.

Igor Labutov, Shashank Srivastava, and Tom Mitchell. 2018. Lia: A natural language programmable personal assistant. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 145–150. Association for Computational Linguistics.

John E Laird, Kevin Gluck, John Anderson, Kenneth D Forbus, Odest Chadwicke Jenkins, Christian Lebiere, Dario Salvucci, Matthias Scheutz, Andrea Thomaz, Greg Trafton, et al. 2017. Interactive task learning. *IEEE Intelligent Systems*, 32(4):6–21.

Vu Le, Sumit Gulwani, and Zhendong Su. 2013. SmartSynth: Synthesizing Smartphone Automation Scripts from Natural Language. In *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys ’13, pages 193–206, New York, NY, USA. ACM.

Tak Yeon Lee, Casey Dugan, and Benjamin B. Bederson. 2017. Towards Understanding Human Mistakes of Programming by Example: An Online User Study. In *Proceedings of the 22nd International Conference on Intelligent User Interfaces*, IUI ’17, pages 257–261, New York, NY, USA. ACM.

Gilly Leshed, Eben M. Haber, Tara Matthews, and Tessa Lau. 2008. CoScripter: Automating & Sharing How-to Knowledge in the Enterprise. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI ’08, pages 1719–1728, New York, NY, USA. ACM.

Toby Jia-Jun Li, Amos Azaria, and Brad A. Myers. 2017a. SUGILITE: Creating Multimodal Smartphone Automation by Demonstration. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI ’17, pages 6038–6049, New York, NY, USA. ACM.

Toby Jia-Jun Li, Jingya Chen, Brandon Canfield, and Brad A. Myers. 2020. Privacy-preserving script sharing in gui-based programming-by-demonstration systems. *Proc. ACM Hum.-Comput. Interact.*, 4(CSCW).

Toby Jia-Jun Li, Igor Labutov, Xiaohan Nancy Li, Xiaoyi Zhang, Wenze Shi, Tom M. Mitchell, and Brad A. Myers. 2018a. APPINITE: A Multi-Modal Interface for Specifying Data Descriptions in Programming by Demonstration Using Verbal Instructions. In *Proceedings of the 2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC 2018)*.

Toby Jia-Jun Li, Igor Labutov, Brad A. Myers, Amos Azaria, Alexander I. Rudnicky, and Tom M. Mitchell. 2018b. Teaching Agents When They Fail: End User Development in Goal-oriented Conversational Agents. In *Studies in Conversational UX Design*. Springer.

Toby Jia-Jun Li, Yuanchun Li, Fanglin Chen, and Brad A. Myers. 2017b. Programming IoT Devices by Demonstration Using Mobile Apps. In *End-User Development*, pages 3–17, Cham. Springer International Publishing.

Toby Jia-Jun Li, Marissa Radensky, Justin Jia, Kirielle Singarajah, Tom M. Mitchell, and Brad A. Myers. 2019. PUMICE: A Multi-Modal Agent that Learns Concepts and Conditionals from Natural Language and Demonstrations. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology (UIST 2019)*, UIST 2019. ACM.

Toby Jia-Jun Li and Oriana Riva. 2018. KITE: Building conversational bots from mobile apps. In *Proceedings of the 16th ACM International Conference on Mobile Systems, Applications, and Services (MobiSys 2018)*. ACM.

Percy Liang. 2016. Learning executable semantic parsers for natural language understanding. *Communications of the ACM*, 59(9):68–76.

Percy Liang, Michael I. Jordan, and Dan Klein. 2013. Learning dependency-based compositional semantics. *Computational Linguistics*, 39(2):389–446.

Henry Lieberman. 2001. *Your wish is my command: Programming by example*. Morgan Kaufmann.

Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, and Percy Liang. 2018a. Reinforcement learning on web interfaces using workflow-guided exploration. In *International Conference on Learning Representations*.

Thomas F Liu, Mark Craft, Jason Situ, Ersin Yumer, Radomir Mech, and Ranjitha Kumar. 2018b. Learning design semantics for mobile apps. In *The 31st Annual ACM Symposium on User Interface Software and Technology*, pages 569–579. ACM.

Richard G. McDaniel and Brad A. Myers. 1999. Getting More out of Programming-by-demonstration. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '99, pages 442–449, New York, NY, USA. ACM.

Brad A. Myers, Amy 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*, pages 1–22. Springer, Cham.

Brad A. Myers and Richard McDaniel. 2001. Sometimes you need a little intelligence, sometimes you need a lot. *Your Wish is My Command: Programming by Example. San Francisco, CA: Morgan Kaufmann Publishers*, pages 45–60.

Chelsea Myers, Anushay Furqan, Jessica Nebolsky, Karina Caro, and Jichen Zhu. 2018. Patterns for how users overcome obstacles in voice user interfaces. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–7.

Sharon Oviatt. 1999. Mutual disambiguation of recognition errors in a multimodel architecture. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 576–583. ACM.

Panupong Pasupat, Tian-Shun Jiang, Evan Liu, Kelvin Guu, and Percy Liang. 2018. Mapping natural language commands to web elements. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4970–4976, Brussels, Belgium. Association for Computational Linguistics.

Panupong Pasupat and Percy Liang. 2015. Compositional Semantic Parsing on Semi-Structured Tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*. ArXiv: 1508.00305.

Alborz Rezazadeh Sereshkeh, Gary Leung, Krish Perumal, Caleb Phillips, Minfan Zhang, Afsaneh Fazly, and Iqbal Mohomed. 2020. Vasta: a vision and language-assisted smartphone task automation system. In *Proceedings of the 25th International Conference on Intelligent User Interfaces*, pages 22–32.

Shashank Srivastava, Igor Labutov, and Tom Mitchell. 2017. Joint concept learning and semantic parsing from natural language explanations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1527–1536.

Shashank Srivastava, Igor Labutov, and Tom Mitchell. 2018. Zero-shot learning of classifiers from natural language quantification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 306–316, Melbourne, Australia. Association for Computational Linguistics.

Frank F Xu, Zhengbao Jiang, Pengcheng Yin, Bogdan Vasilescu, and Graham Neubig. 2020. Incorporating external knowledge through pre-training for natural language to code generation. *arXiv preprint arXiv:2004.09015*.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 440–450, Vancouver, Canada. Association for Computational Linguistics.