Programming by Demonstration with Situated Semantic Parsing

Yoav Artzi*, **Maxwell Forbes***, **Kenton Lee*** **& Maya Cakmak**
Department of Computer Science & Engineering, University of Washington
185 NE Stevens Way, Seattle, WA 98195

Introduction

Programming by Demonstration (PbD) is an approach to programming robots by demonstrating the desired behavior (Billard et al. 2008). Speech is a natural, hands-free way to augment demonstrations with control commands that guide the PbD process. However, existing speech interfaces for PbD systems rely on ad-hoc, predefined command sets that are rigid and require user training (Weiss et al. 2009; Akgun et al. 2012; Cakmak and Takayama 2014). Instead, we aim to develop flexible speech interfaces to accommodate user variations and ambiguous utterances. To that end, we propose to use a situated semantic parser that jointly reasons about the user's speech and the robot's state to resolve ambiguities. In this paper, we describe this approach and compare its utility to a rigid speech command interface.

Approach

Domain

We extend the work of Cakmak and Takayama (2014) in which users program actions on a PR2 robot by physically moving the robot's arms and using predefined speech commands to regulate the interaction (Fig. 1(a)). Learned skills are represented as a sparse sequence of keyframes (6-dimensional configurations of the two end-effectors and gripper states) that the robot iterates through (Akgun et al. 2012). Each skill is programmed directly through a single demonstration. Speech commands are used for changing the robot state (arm stiff/relaxed, gripper open/closed), creating and switching between skills, adding keyframes to the current skill, executing and clearing the current skill and undoing. Fig. 1(b) lists all commands. The robot responds to each command verbally and with a head gesture.

Situated Semantic Parsing

Our work aims to map unconstrained spoken instructions to known commands. We frame this problem as a semantic parsing problem, where the meaning of instructions is represented with lambda calculus expressions. We consider both the spoken sentence and the robot state to jointly reason about the logical form representing the meaning of a sentence and the execution of this logical form to a robot command. Following Artzi and Zettlemoyer (2013b), we use Combinatory Categorial Grammar (CCG) a linguistically-motivated formalism for modeling a wide range of language phenomena (Steedman 1996; 2000).

(b) TEST MICROPHONE
RELEASE/HOLD RIGHT/LEFT ARM
OPEN/CLOSE RIGHT/LEFT HAND
CREATE SKILL
SAVE POSE
EXECUTE SKILL
CLEAR SKILL
NEXT/PREVIOUS SKILL
UNDO LAST COMMAND

Figure 1: (a) Participant programming a towel folding skill on the PR2 robot. (b) Full list of speech commands for interacting with the robot in our baseline condition.

Let $\mathcal{S}$ be a set of states, $\mathcal{X}$ the set of all possible sentences and $\mathcal{E}$ the space of executions, which are $\mathcal{S} \rightarrow \mathcal{S}$ functions. Each $s \in \mathcal{S}$ is a fully specified state of the robot and the interaction, including the status of each $hand$ ($open$/$close$), stiffness of each arm ($released$/$held$), and a record of properties that were changed most recently. The set of properties is $\{arm, hand, left, right, skill, pose\}$. For example, if the last object to change is the left arm, the state will include the mappings $left \rightarrow left\text{-}arm$ and $arm \rightarrow left\text{-}arm$. An execution e is one of the original commands listed in Fig. 1(b) for which the robot has a known subroutine and a response. Given a state $s \in \mathcal{S}$ and a sentence $x \in \mathcal{X}$, we aim to find the execution $e \in \mathcal{E}$ intended by x.

Given a sentence x, we find the most likely execution using a situated linear model. Let $\mathcal{Y}$ be the set of all CCG parse trees and $\mathcal{Z}$ be the set of all logical forms. Given a sentence $x \in \mathcal{X}$, parse tree $y \in \mathcal{Y}$, logical form $z \in \mathcal{Z}$ and an execution $e \in \mathcal{E}$, let a derivation d be a tuple $\langle x, y, z, e \rangle$. In d, e is an execution of the logical form z, which is contained in the root of the parse tree y generated from the sentence x and state s. $\mathcal{D}$ is the set of all derivations. Let $\text{GEN}(x, s) \subset \mathcal{D}$ be the set of all derivations given a sentence x and a state s. The optimal derivation for a sentence x and state s is $d^*(x, s) = \arg\max_{d \in \text{GEN}(x,s)} \theta \cdot \phi(x, s, d)$, where $\phi(x, s, d)$ is a feature vector and $\theta \in \mathbb{R}^d$ a weight vector. The optimal execution e^* is extracted from $d^*(x, s)$. For example, the command *close right hand* is mapped to the logical form $\lambda a.close(a) \wedge patient(a, \mathcal{A}(\lambda x.right(x) \wedge hand(x)))$ and the execution $close\text{-}right\text{-}hand$.

*The first three authors contributed equally to this paper.

Jointly reasoning over semantic meaning and execution allows our system to correctly execute underspecified commands. Consider the command *close* and its logical form $\lambda a.close(a)$, which only specifies the action type and corresponds to multiple executions, including *close-left-hand*, *close-right-hand*, *close-right-arm*, and even *close-skill*. There are two sources of information that our system uses to select the correct execution. First, in our system only the *hand* object affords the *close* operation, which indicates that the system should ignore both *close-right-arm* and *close-skill*. Second, the robot state allows the system to estimate object saliency and the impact of an action. For example, if the state indicates that the left hand is closed and the right is open, the parser will prefer *close-right-hand* over *close-left-hand*. Alternatively, if both hands are open, but the left limb was changed more recently (by the user or the robot), *close-left-hand* will be preferred.

Implementation

Our PbD implementation is based on the open source implementation of Cakmak and Takayama (2014). To implement a situated semantic parser, we used UW SPF (Artzi and Zettlemoyer 2013a). Following Lee et al. (2014) we manually designed a CCG lexicon. We initialized the lexicon with the entries required to correctly parse the natural language commands used by Cakmak and Takayama (2014). We then used Wordnet (Fellbaum 2010) to automatically expand the lexicon and manually pruned the expanded set of lexical entries. We also added lexical entries to handle coordination (e.g. *close left and right hand*). Our model includes features that indicate lexical entry usage, state-execution correspondence, and various properties of the logical form. Model weights are set manually. We use the Google speech recognizer and consider phoneme distance to generate lexical entries on the fly to overcome common speech recognition errors.

Evaluation

We evaluate our approach with the experimental setup used by Cakmak and Takayama (2014) in the *Video* condition.

Comparison with Expert Users

We first evaluate our system with three expert users (developers of the system). Each user programmed four skills in two conditions: using exact commands with the system of Cakmak and Takayama (2014) (baseline) and using the semantic parsing interface. In the latter condition, the experts were able to exploit the joint reasoning over language and robot state, to use shorter, more implicit commands. On average, 1.6 words were used per utterance with the semantic parser, compared to 2.4 words per utterance in the baseline condition ($t(84) = 5.21, p < .01$). However, this did not result in an observed improvement in completion time. Additionally, the experts used fewer utterances to program each skill, but this difference was not significant. For comparison, Fig. 2 shows results for novice users in Cakmak and Takayama (2014) and our expert results for Skill #3.

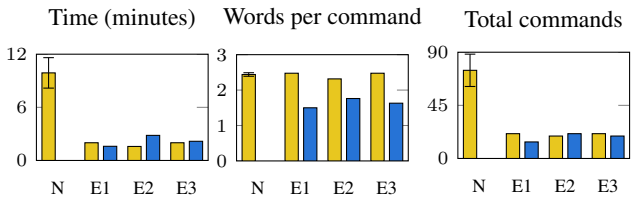

Figure 2: Programming time, words per command and total number of commands for Skill #3, for novice users (N) (mean and standard deviation) and experts (E1, E2 and E3) using the baseline (■) and the semantic parsing (■) interfaces.

Comparison with Novice Users

We conducted a pilot user study with 8 participants. Our participants mostly used the exact commands as in the instructions and rarely exploited the flexibility of the semantic parser. Potential factors contributing to this finding are the noisy speech recognizer, prior expectations about natural language interfaces and the instructional material, which was identical to the original baseline. Additionally, speech recognition errors were more common in the semantic parsing condition, since we did not limit the language that could be used.

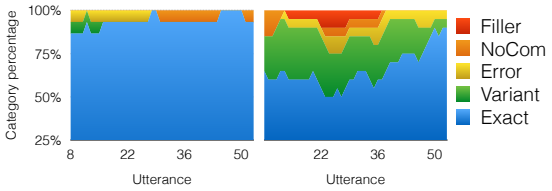

Figure 3: Frequency of utterance categories for the first 60 utterances for *P1* (left) and *P2* (right). A moving average over 15 utterances is displayed for readability.

We partially coded the sessions of two participants (*P1* and *P2*) for error analysis. The first 60 utterances of each session were manually classified as: (*a*) *Exact*: correctly recognized exact match to one of the original commands, (*b*) *Variant*: correctly recognized variation of one of the original commands, (*c*) *Error*: no command or the wrong command recognized, (*d*) *NoCom*: input that did not correspond to any of the possible commands, or (*e*) *Filler*: utterances not intended as commands. The language used by *P1* was typical of our participant pool, which mostly consisted of computer science students. Participants often strictly adhered to the original commands that were provided to them as examples. An exception to this trend was *P2*, who showed a wider variation in language use, which we speculate is due to different expectations of system abilities.

Discussion

Our preliminary study suggests a few directions for future work. To allow users to fully take advantage of the system, we need to design instructions to better support rich interactions. Speech recognition can be improved by considering domain knowledge, such as the set of existing objects and actions. Finally, the data obtained from user interactions could be used to improve our system over time, for example, to learn a better language model or induce object affordances, as described by Artzi and Zettlemoyer (2011).

References

Akgun, B.; Cakmak, M.; Jiang, K.; and Thomaz, A. L. 2012. Keyframe-based learning from demonstration. *International Journal of Social Robotics* 343–355.

Artzi, Y., and Zettlemoyer, L. 2011. Bootstrapping semantic parsers from conversations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Artzi, Y., and Zettlemoyer, L. 2013a. UW SPF: The University of Washington Semantic Parsing Framework.

Artzi, Y., and Zettlemoyer, L. 2013b. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics* 1(1):49–62.

Billard, A.; Calinon, S.; Dillmann, R.; and Schaal, S. 2008. Robot Programming by Demonstration. In *Handbook of Robotics*. Springer. chapter 59.

Cakmak, M., and Takayama, L. 2014. Teaching people how to teach robots: The effect of instructional materials and dialog design. In *Proceedings of the ACM/IEEE International Conference on Human-robot Interaction*.

Fellbaum, C. 2010. *WordNet*. Springer.

Lee, K.; Artzi, Y.; Dodge, J.; and Zettlemoyer, L. 2014. Context-dependent semantic parsing for time expressions. In *Proceedings of the Conference of the Association for Computational Linguistics*.

Steedman, M. 1996. *Surface Structure and Interpretation*. The MIT Press.

Steedman, M. 2000. *The Syntactic Process*. The MIT Press.

Weiss, A.; Igelsboeck, J.; Calinon, S.; Billard, A.; and Tscheligi, M. 2009. Teaching a humanoid: A user study on learning by demonstration with hoap-3. In *IEEE Symposium on Robot and Human Interactive Communication (RO-MAN)*, 147–152.