Learning to Interpret Natural Language Instructions

Monica Babeş-Vroman*, James MacGlashan*, Ruoyuan Gao†, Kevin Winner*, Richard Adjogah*, Marie desJardins*, Michael Littman† and Smaranda Muresan++

*Department of Computer Science and Electrical Engineering
  University of Maryland, Baltimore County
+Computer Science Department, Rutgers University
++School of Communication and Information, Rutgers University

NAACL-2012 WORKSHOP
From Words to Actions: Semantic Interpretation in an Actionable Context
Outline

• Motivation and Problem
• Our approach
• Pilot study
• Conclusions and future work
Motivation

Bring me the red mug that I left in the conference room
Motivation
Motivation
Problem

Train an artificial agent to learn to carry out complex multistep tasks specified in natural language.
Another Example of a Task of pushing an object to a room.

Abstract task: move obj to color room

- move square to red room
- move star to green room
- go to green room

Push the square to the red room.

Push the star into the green room

Go to side green.

Move the box to the coral room.

Push the star into the teal room.

End at the green.

Take the rectangle to the red zone.

Take the star to the green room.

Go into the green room.
Outline

• What is the problem?
• Our approach
• Pilot study
• Conclusions and future work
Training Data:

Object-oriented Markov Decision Process (OO-MDP)

[Diuk et al., 2008]
“Push the star into the teal room”
Our Approach

"Push the star into the teal room"

Semantic Parsing

push(star1, room1)
P1(room1, teal)

Task Learning from Demonstrations

F1 F2 F3 F4 F5 F6

Task Abstraction
Our Approach

“Push the star into the teal room”

Semantic Parsing

Task Learning from Demonstrations

Task Abstraction

push(star1, room1)
P1(room1, teal)

“teal” ➔ teal
P1 ➔ color
“push” ➔ objToRoom

F2 F6
Our Approach

“Push the star into the teal room”

Semantic Parsing

Task Abstraction

Task Learning from Demonstrations
Task Learning from Demonstration

= Inverse Reinforcement Learning (IRL)

• But first let’s see Reinforcement Learning Problem
Task Learning From Demonstrations

• Inverse Reinforcement Learning
Task Learning From Demonstrations

• Inverse Reinforcement Learning
Task Learning From Demonstrations

• Inverse Reinforcement Learning
• Maximum Likelihood IRL


**Assumption:** Reward function is a linear combination of a known set of features. (e.g., 🎉😊⭐⭐⭐)

\[ R(s, a) = \sum_i \theta_i \varphi_i(s, a) = \theta \times \varphi(s, a) \]

**Goal:** Find \( \theta \) to maximize the probability of the observed trajectories.

(s,a): (state, action) pair
R: reward function
\( \phi \): feature vector
\( \pi \): policy
Pr(STRU): probability of demonstrations
Our Approach

“Push the star into the teal room”

Semantic Parsing

Task Abstraction

Task Learning from Demonstrations
Semantic Parsing

• Lexicalized Well-Founded Grammars (LWFG)
  [Muresan, 2006; 2008; 2012]
Semantic Parsing

• Lexicalized Well-Founded Grammars (LWFG)
  [Muresan, 2006; 2008; 2012]

Grammar

$$\text{NP} (w_1, \underline{\text{X1}}) \rightarrow \text{Adj} (w_1, \underline{\text{X1}}), \text{NP} (w_2, \underline{\text{X1}}): \Phi_i$$
$$\text{NP} (w_1, \underline{\text{X1}}) \rightarrow \text{Det} (w_1, \underline{\text{X1}}), \text{NP} (w_2, \underline{\text{X1}}): \Phi_i$$
...

Empty Semantic model

```
P1
```

uninstantiated variable

```
teal
```

```
room1
```

```
X1.is=teal, X.P1=X1, X.isa=room
```

```
PARSING
```

```
teal room
```

```
``
Semantic Parsing

• Lexicalized Well-Founded Grammars (LWFG)
  [Muresan, 2006; 2008; 2012]

Grammar

NP (w, ) → Adj (w₁, ), NP (w₂, ): Φᵢ
NP (w, ) → Det (w₁, ), NP (w₂, ): Φᵢ
...

Syntax/Semantics/Ontological Constraints

teal room

X₁.is=green, X.color=X₁, X.isa=room

Semantic model
(ontology)

room1

color
green
Learning LWFG

- [Muresan and Rambow 2007; Muresan, 2010; 2011]
Currently learning is done offline
No assumption of existing semantic model

**Small Training Data**

**Representative Examples**

NP (loud noise)
NP (the noise)
PP ...

**Generalization Set**

NP (loud noise)
NP (the noise)
NP (loud clear noise)
NP (the loud noise)
...

**RELATIONAL LEARNING MODEL**

LWFG

NP (w) → Adj (w₁), NP (w₂): Φᵢ
NP (w) → Det (w₁), NP (w₂): Φᵢ
...

Using LWFG for Semantic Parsing

• Obtain representation with same underlying meaning despite different syntactic forms
  “go to the green room” vs “go to the room that is green”

  – Dynamic Semantic Model
  Can start with an empty semantic model
  Augmented based on feedback from the TA component
  push ➔ blockToRoom, teal ➔ green
Our Approach

“Push the star into the teal room”

Semantic Parsing

Task Abstraction

Task Learning from Demonstrations
Task Abstraction

• Creates an abstract task to represent a set of similar *grounded* tasks
  – Specifies goal conditions and reward function in terms of OO-MDP propositional functions

• Combines information from SP and IRL and provides feedback to both
  – Provides SP domain specific semantic information
  – Provides IRL relevant features
Some grounded tasks are similar in logical goal conditions, but different in object references and facts about those referenced objects.
Abstract Task Definition

- Define task conditions as a partial first order logic expression

\[ \exists o \in Object, l \in Location \ \text{ObjectIn}(o,l) \]

- The expression takes propositional function parameters to complete it

\[ \exists o \in Object, l \in Location \ \text{ObjectIn}(o,l) \land \text{isYellow}(o) \land \text{isStar}(o) \land \text{isGreen}(l) \]
System Structure

- Semantic Parsing
- Inverse Reinforcement Learning (IRL)

Task Abstraction
Pilot Study

Unigram Language Model

**Data:** \((S_i, T_i)\).

**Hidden Data:** \(z_{ji} : \Pr(R_j \mid (S_i, T_i))\).

**Parameters:** \(x_{kj} : \Pr(w_k \mid R_j)\).

**Algorithm 3 EM Model 0**

**Input:** Demonstrations \(\{(S_1, T_1), \ldots, (S_N, T_N)\}\), number of reward functions \(J\), size of vocabulary \(K\).

**Initialize:** \(x_{11}, \ldots, x_{JK}\), randomly.

**repeat**

**E Step:** Compute
\[
    z_{ji} = \frac{\Pr(R_j)}{\Pr(S_i, T_i)} \Pr(T_i \mid R_j) \prod_{w_k \in S_i} x_{kj}.
\]

**M step:** Compute
\[
    x_{kj} = \frac{1}{X} \frac{\sum_{w_k \in S_i} \Pr(R_j \mid S_i) + \epsilon}{\sum_i N(S_i) z_{ji} + \epsilon}.
\]

until target number of iterations completed.
Pilot Study

Unigram Language Model

Data: \((S_i, T_i)\).

Hidden Data: \(z_{ji} : \text{Pr}(R_j | (S_i, T_i))\).

Parameters: \(x_{kj} : \text{Pr}(w_k | R_j)\).

---

Algorithm 3 EM Model 0

Input: Demonstrations \(\{(S_1, T_1), \ldots, (S_N, T_N)\}\), number of reward functions \(J\), size of vocabulary \(K\).

Initialize: \(x_{11}, \ldots, x_{JK}\), randomly.

repeat

E Step: Compute

\[
z_{ji} = \frac{\text{Pr}(R_j)}{\text{Pr}(S_i, T_i)} \cdot \text{Pr}(T_i | R_j) \prod_{w_k \in S_i} x_{kj}.
\]

M step: Compute

\[
x_{kj} = \frac{1}{X} \frac{\sum_{w_k \in S_i} \text{Pr}(R_j | S_i)}{\sum_i N(S_i) z_{ji} + \epsilon}.
\]

until target number of iterations completed.

---

\[\text{Pr(“push”|R1)} \ldots \text{Pr(“room”|R1)}\]
\[\text{Pr(“push”|R2)} \ldots \text{Pr(“room”|R2)}\]
Pilot Study

New Instruction, $S_N$: “Go to the green room.”

$$\Pr(S_N \mid R_1) = \prod_{w_k \in S_N} \Pr(w_k \mid R_1) = 8.6 \times 10^{-7},$$

$$\Pr(S_N \mid R_2) = \prod_{w_k \in S_N} \Pr(w_k \mid R_2) = 4.1 \times 10^{-4}$$
Pilot Study

New Instruction, $S'N$: “Go with the star to green.”

\[
\begin{align*}
\Pr(S'_N \mid R_1) &= \prod_{w_k \in S'_N} \Pr(w_k \mid R_1) = 8.3 \times 10^{-7}, \\
\Pr(S'_N \mid R_2) &= \prod_{w_k \in S'_N} \Pr(w_k \mid R_2) = 2.1 \times 10^{-5}
\end{align*}
\]
Outline

• What is the problem?
• Our approach
• Pilot study
• Conclusions and future work
Conclusions

- Proposed new approach for training an artificial agent to learn to carry out complex multistep tasks specified in natural language, from pairs of instructions and demonstrations
- Showed pilot study with a simplified model
Current/Future Work

- SP, TA and IRL integration
- High-level multistep tasks
- Probabilistic LWFGs
  - Probabilistic domain (semantic) model
  - Learning/Parsing algorithms with both hard and soft constraints
- TA – add support for hierarchical task definitions that can handle temporal conditions
- IRL – find a partitioning of the execution trace where each partition has its own reward function
Thank you!