Environment-Driven Lexicon Induction for High-Level Instructions

Dipendra K. Misra[*] *dkm@cs.cornell.edu*

Kejia Tao[*] *kt454@cornell.edu*

Percy Liang[**] *pliang@cs.stanford.edu*

Ashutosh Saxena[*] *asaxena@cs.cornell.edu*

[*] Department of Computer Science, Cornell University
[**]Department of Computer Science, Stanford University

Abstract

We focus on the task of interpreting complex natural language instructions to a robot, in which we must ground high-level commands such as *microwave the cup* to low-level actions such as grasping. Previous approaches that learn a lexicon during training have inadequate coverage at test time, and pure search strategies cannot handle the exponential search space. We propose a new hybrid approach that leverages the environment to induce new lexical entries at test time, even for new verbs. Our semantic parsing model jointly reasons about the text, logical forms, and environment over multi-stage instruction sequences. We introduce a new dataset and show that our approach is able to successfully ground new verbs such as *distribute, mix, arrange* to complex logical forms, each containing up to four predicates.

1 Introduction

The task of mapping natural language instructions to actions for a robot has been gaining momentum in recent years (Artzi and Zettlemoyer, 2013; Tellex et al., 2011; Misra et al., 2014; Bollini et al., 2011; Guadarrama et al., 2013; Matuszek et al., 2012b; Fasola and Mataric, 2013). We are particularly interested in instructions containing verbs such as "*microwave*" denoting high-level concepts, which correspond to more than 10 low-level symbolic actions such as `grasp`. In this setting, it is common to find new verbs requiring new concepts at test time. For example, in Figure 1, suppose that we have never seen the verb "*fill*". Can we impute the correct interpretation, and moreover seize the opportunity to learn what "*fill*" means in a way that generalizes to future instructions?

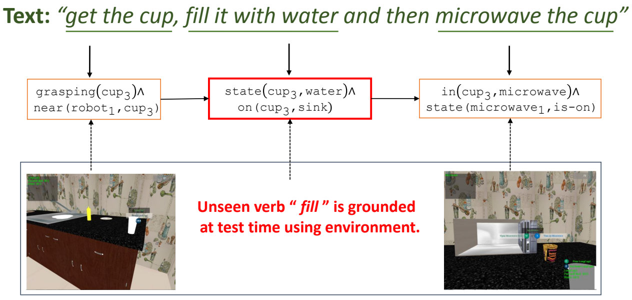
Text: "get the cup, fill it with water and then microwave the cup"
grasping(cup$_3$)∧ near(robot$_1$,cup$_3$)
state(cup$_3$,water)∧ on(cup$_3$,sink)
in(cup$_3$,microwave)∧ state(microwave$_1$,is-on)
Unseen verb " *fill* " is grounded at test time using environment.

Figure 1: A lexicon learned on the training data cannot possibly cover all the verb-concept mappings needed at test time. Our algorithm learns the meaning of new verbs (e.g., "*fill*") using the environment context.

Previous work in semantic parsing handles lexical coverage in one of two ways. Kwiatkowski et al. (2010) induces a highly constrained CCG lexicon capable of mapping words to complex logical forms, but it would have to skip new words (which in Figure 1 would lead to microwaving an empty cup). Others (Berant and Liang, 2014) take a freer approach by performing a search over logical forms, which can handle new words, but the logical forms there are much simpler than the ones we consider.

In this paper, we present an hybrid approach that uses a lexicon to represent complex concepts but also strongly leverages the environment to guide the search space. The environment can provide helpful cues in several ways:

- Only a few environments are likely for a given scenario—e.g., the text is unlikely to ask the robot to microwave an empty cup or put books on the floor.
- The logical form of one segment of text constrains that of the next segment—e.g., the text is unlikely to ask the robot to pick a cup and then put it back immediately in the same spot.

We show that this environment context provides

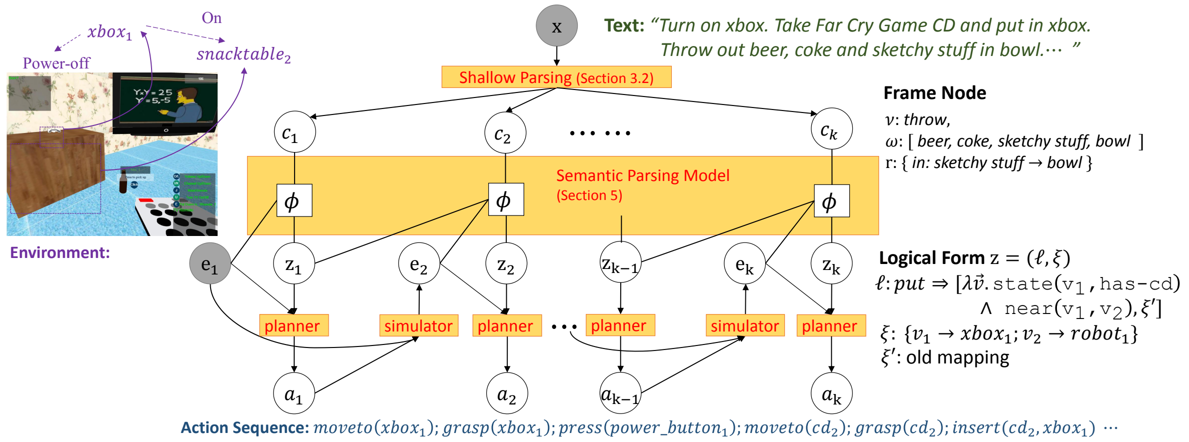
On
xbox₁
Power-off
snacktable₂
Environment:
x
Text: "Turn on xbox. Take Far Cry Game CD and put in xbox. Throw out beer, coke and sketchy stuff in bowl.⋯ "
Shallow Parsing (Section 3.2)
c₁
c₂
c_k
Frame Node
ν: throw,
ω: [beer, coke, sketchy stuff, bowl]
r: { in: sketchy stuff → bowl }
Semantic Parsing Model (Section 5)
ϕ
e₁
z₁
e₂
z₂
z_{k−1}
e_k
z_k
Logical Form z = (ℓ, ξ)
ℓ: put ⇒ [λv⃗. state(v₁, has-cd) ∧ near(v₁, v₂), ξ′]
ξ: {v₁ → xbox₁; v₂ → robot₁}
ξ′: old mapping
planner
simulator
a₁
a₂
a_{k−1}
a_k
Action Sequence: moveto(xbox₁); grasp(xbox₁); press(power_button₁); moveto(cd₂); grasp(cd₂); insert(cd₂, xbox₁) ⋯

Figure 2: Graphical model overview: we first deterministically shallow parse the text x into a control flow graph consisting of shallow structures $\{c_i\}$. Given an initial environment e_1, our semantic parsing model maps these frame nodes to logical forms $\{z_i\}$ representing the postconditions. From this, a planner and simulator generate the action sequences $\{a_i\}$ and resulting environments $\{e_i\}$.

a signal for inducing new lexical entries that map previously unseen verbs to novel concepts. In the example in Figure 1, the algorithm learns that microwaving an empty cup is unlikely and this suggests that the verb "*fill*" must map to actions that end up making the cup not empty.

Another contribution of this paper is using postconditions as logical forms rather than actions, as in previous work (Artzi and Zettlemoyer, 2013; Misra et al., 2014). Postconditions not only reduce the search space of logical forms, but are also a more natural representation of verbs. We define a conditional random field (CRF) model over postconditions, and use a planner to convert postconditions into action sequences and a simulator to generate new environments.

At test time, we use the lexicon induced from the training data, but also perform an environment-guided search over logical forms to induce new lexical entries on-the-fly. If the predicted action sequence uses a new lexical entry generated by the search, it is added to the lexicon, where it can be reused in subsequent test examples.

We evaluate our algorithm on a new corpus containing text commands for a household robot. The two key findings of our experiments are: First, the environment and task context contain enough information to allow us to learn lexical entries for new verbs such as "*distribute*" and "*mix*" with complex semantics. Second, using both lexical entries generated by a test-time search and those from the lexicon induced by the training data outperforms the two individual approaches. This suggests that environment context can help alleviate the problem of having a limited lexicon for grounded language acquisition.

2 Problem Statement

At training time, we are given a set of examples $D = \{(x^{(m)}, e^{(m)}, a^{(m)}, \pi^{(m)})\}_{m=1}^{M}$, where $x^{(m)}$ is a text containing natural language instructions, $e^{(m)}$ is an initial environment, $a^{(m)}$ is a human-annotated sequence of actions, and $\pi^{(m)}$ specifies a monotonic alignment between segments of $x^{(m)}$ and segments of $a^{(m)}$. For example, given words $x^{(m)} = x_1x_2$ and $a^{(m)} = a_1a_2a_3$, $\pi^{(m)}$ might specify that x_1 aligns to a_1a_2 and x_2 aligns to a_3.

At test time, given a sequence of text-environment pairs as input $\{(x^{(n)}, e^{(n)})\}_{n=1}^{N}$, we wish to generate a sequence of actions $a^{(n)}$ for each input pair. Note that our system is allowed to use information about one test example to improve performance on subsequent ones. We evaluate a system on its ability to recover a human-annotated sequence of actions.

3 Approach Overview

Figure 2 shows our approach for mapping text x to actions $a_{1:k}$ given the initial environment e_1.

3.1 Representation

We use the following representation for the different variables in Figure 2.

Environment. An environment e_i is represented by a graph whose nodes are objects and edges represent spatial relations between these objects. We consider five basic spatial relations: `near`, `grasping`, `on`, `in` and `below`. Each object has an instance ID (e.g., $\texttt{book}_9$), a category name (e.g., *chair, xbox*), a set of properties such as *graspable, pourable* used for planning and a set of boolean states such as `has-water`, `at-channel3`, whose values can be changed by robot actions. The robot is also an object in the environment. For example, the objects $\texttt{xbox}_1, \texttt{snacktable}_2$, are two objects in e_1 in Figure 2 with relation on between them.

Postconditions. A postcondition is a conjunction of atoms or their negations. Each atom consists of either a spatial relation between two objects (e.g., $\texttt{on}(\texttt{book}_9, \texttt{shelf}_3)$) or a state and a value (e.g., $\texttt{state}(\texttt{cup}_4, \texttt{has-water})$). Given an environment e, the postcondition evaluates to true or false.

Actions. Each action in an action sequence a_i consists of an action name with a list of arguments (e.g., $\texttt{grasp}(\texttt{xbox}_1)$). The action name is one of 15 values ($\texttt{grasp}, \texttt{moveto}, \texttt{wait}$, etc.), and each argument is either an object in the environment (e.g., $\texttt{xbox}_1$), a spatial relation (e.g., `in` for $\texttt{keep}(\texttt{ramen}_2, \texttt{in}, \texttt{kettle}_1)$), or a postcondition (e.g., for $\texttt{wait}(\texttt{state}(\texttt{kettle}_1, \texttt{boiling}))$).

Logical Forms. The logical form z_i is a pair (ℓ, ξ) containing a lexical entry ℓ and a mapping ξ. The lexical entry ℓ contains a parameterized postcondition such as $\lambda\vec{\texttt{v}}.\texttt{grasping}(\texttt{v}_1, \texttt{v}_2)$ $\wedge\neg\texttt{near}(\texttt{v}_3, \texttt{v}_2)$, and ξ maps the variables $\vec{\texttt{v}}$ to objects in the environment. Applying the parameterized postcondition on ξ yields a postcondition; note that a postcondition can be represented by different logical forms. A lexical entry contains other information which are used for defining features, which is detailed in Section 4.

Control Flow Graphs. Following previous work (Tellex et al., 2011; Misra et al., 2014), we convert the text x to a shallow representation. The particular representation we choose is a *control flow graph*, which encodes the sequential relation between atomic segments in the text. Figure 3 shows the control flow graph for an example text. In a *control flow graph*, each node is either a frame node or a conditional node. A frame node represents a single clause (e.g., "*change the channel to a movie*") and has at most one successor node. Specifically, a frame node consists of a verb ν (e.g., *arrange*, *collect*), a set of object descriptions $\{\omega_i\}$ which are the arguments of the verb (e.g., *the guinness book*, *movie channel*), and spatial relations r between the arguments (e.g., *between*, *near*). The object description ω is either an anaphoric reference (such as "*it*") or a tuple containing the main noun, associated modifiers, and relative clauses.

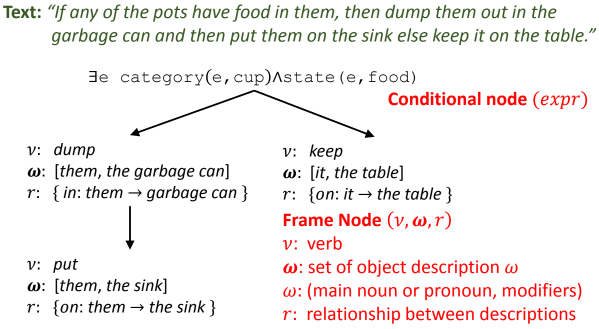
Text: "If any of the pots have food in them, then dump them out in the garbage can and then put them on the sink else keep it on the table."
∃e category(e,cup)∧state(e,food)
Conditional node (expr)
ν: dump
ω: [them, the garbage can]
r: { in: them → garbage can }
ν: put
ω: [them, the sink]
r: {on: them → the sink }
ν: keep
ω: [it, the table]
r: {on: it → the table }
Frame Node (ν, ω, r)
ν: verb
ω: set of object description ω
ω: (main noun or pronoun, modifiers)
r: relationship between descriptions

Figure 3: We deterministically parse text into a shallow structure called a control flow graph.

A conditional node contains a logical postcondition with at most one existentially quantified variable (in contrast to a frame node, which contains natural language). For example, in Figure 3 the conditional node contains the expression corresponding to the text *"if any of the pots has food"* There are two types of conditional nodes: branching and temporal. A branching conditional node represents an "*if*" statement and has two successor nodes corresponding to whether the condition evaluates to true or false in the current environment. A temporal conditional node represents an "*until*" statement and waits until the condition is false in the environment.

3.2 Formal Overview

Shallow Parsing. We deterministically convert the text x into its *control flow graph* G using a set of manual rules applied on its constituency parse tree from the Stanford parser (Klein and Manning, 2003). Conditionals in our dataset are simple and can be converted into postconditions directly using a few rules, unlike the action verbs (e.g., "*fill*"), which is the focus of this paper. The details of our shallow parsing procedure is described in the appendix.

Given an environment e_1, G is reduced to a single sequence of frame nodes $c_1, \dots, c_k$, by evaluating all the branch conditionals on e_1.

Semantic Parsing Model. For each frame node c_i and given the current environment e_i, the semantic parsing model (Section 5) places a distribution over logical forms z_i. This logical form z_i represents a postcondition on the environment after executing the instructions in c_i.

Planner and Simulator. Since our semantic representations involve postconditions but our model is based on the environment, we need to connect the two. We use planner and a simulator that together specify a deterministic mapping from the current environment e_i and a logical form z_i to a new environment e_{i+1}. Specifically, the planner takes the current environment e_i and a logical form z_i and computes the action sequence $a_i = \mathit{planner}(e_i, z_i)$ for achieving the post condition represented by z_i.[1] The simulator takes the current environment e_i and an action sequence a_i and returns a new environment $e_{i+1} = \mathit{simulator}(e_i, a_i)$.

4 Anchored Verb Lexicons

Like many semantic parsers, we use a lexicon to map words to logical forms. Since the environment plays an central role in our approach, we propose an *anchored verb lexicon*, in which we store additional information about the environment in which lexical entries were previously used. We focus only on verbs since they have the most complex semantics; object references such as "*cup*" can be mapped easily, as described in Section 5.

More formally, an anchored verb lexicon Λ contains lexical entries ℓ of the following form: $[\nu \Rightarrow (\lambda\vec{v}.S, \xi)]$ where, ν is a verb, S is a postcondition with free variables $\vec{v}$, and ξ is a mapping of these variables to objects. An example lexical entry is: $[\mathit{pour} \Rightarrow (\lambda v_1 v_2 v_3.S, \xi)]$, where:

$$S = \mathtt{grasping}(\mathtt{v}_1, \mathtt{v}_2) \wedge \mathtt{near}(\mathtt{v}_1, \mathtt{v}_3) \wedge \neg\mathtt{state}(\mathtt{v}_2, \mathtt{milk}) \wedge \mathtt{state}(\mathtt{v}_3, \mathtt{milk})$$

$$\xi = \{\mathtt{v}_1 \rightarrow \mathtt{robot}_1, \mathtt{v}_2 \rightarrow \mathtt{cup}_1, \mathtt{v}_3 \rightarrow \mathtt{bowl}_3\} \text{ (anchoring)}$$

As Table 1 shows, a single verb will in general have multiple entries due to a combination of polysemy and the fact that language is higher-level than postconditions.

Advantages of Postconditions. In contrast to previous work (Artzi and Zettlemoyer, 2013; Misra et al., 2014), we use postconditions instead of action sequence for two main reasons. First, postconditions generalize better. To illustrate this, consider the action sequence for the simple task of filling a cup with water. At the time of learning the lexicon, the action sequence might correspond to using a tap for filling the cup while at test time, the environment may not have a tap but instead have a pot with water. Thus, if the lexicon maps to action sequence, then it will not be applicable at test time whereas the postcondition $\mathtt{state}(\mathtt{z}_1, \mathtt{water})$ is valid in both cases. We thus shift the load of inferring environment-specific actions onto planners and use postconditions for representation, which better captures the semantics of verbs.

Table 1: Some lexical entries for the verb "*turn*"

Sentence Context	**Lexical entry** $[turn \Rightarrow (\lambda\vec{v}.S, \xi)]$
"*turn on the TV*"	$\mathtt{state}(\mathtt{v}_1, \mathtt{is\text{-}on}) \wedge \mathtt{near}(\mathtt{v}_2, \mathtt{v}_1)$ $\xi : \mathtt{v}_1 \rightarrow \mathtt{tv}_1, \mathtt{v}_2 \rightarrow \mathtt{robot}_1$
"*turn on the right back burner*"	$\mathtt{state}(\mathtt{v}_1, \mathtt{fire3}) \wedge \mathtt{near}(\mathtt{v}_2, \mathtt{v}_1)$ $\xi : \mathtt{v}_1 \rightarrow \mathtt{stove}_1, \mathtt{v}_2 \rightarrow \mathtt{robot}_1$
"*turn off the water*"	$\neg\mathtt{state}(\mathtt{v}_1, \mathtt{tap\text{-}on})$ $\xi : \mathtt{v}_1 \rightarrow \mathtt{sink}_1$
"*turn the television input to xbox*"	$\mathtt{state}(\mathtt{v}_1, \mathtt{channel6}) \wedge \mathtt{near}(\mathtt{v}_1, \mathtt{v}_2)$ $\xi : \mathtt{v}_1 \rightarrow \mathtt{tv}_1, \mathtt{v}_2 \rightarrow \mathtt{xbox}_1$

Second, because postconditions are higher-level, the number of atoms needed to represent a verb is much less than the corresponding number of actions. For example, the text *"microwave a cup"*, maps to action sequence with 10–15 actions, the postcondition only has two atoms: $\mathtt{in}(\mathtt{cup}_2, \mathtt{microwave}_1) \wedge \mathtt{state}(\mathtt{microwave}, \mathtt{is\text{-}on})$. This makes searching for new logical forms more tractable.

Advantages of Anchoring. Similar to the VEIL templates of Misra et al. (2014), the free variables $\vec{v}$ are associated with a mapping ξ to concrete objects. This is useful for resolving ellipsis. Suppose the following lexical entry was created at training time based on the text "*throw the drinks in the trash bag*":

$[\ell$: *throw* $\Rightarrow \lambda xyz.S(x, y, z)]$, where

$$S = \mathtt{in}(\mathtt{x}, \mathtt{y}) \wedge \neg\mathtt{grasping}(\mathtt{z}, \mathtt{x}) \wedge \neg\mathtt{state}(\mathtt{z}, \mathtt{closed})$$

$$\xi = \{\mathtt{x} \rightarrow \mathtt{coke}_1, \mathtt{y} \rightarrow \mathtt{garbageBin}_1, \mathtt{z} \rightarrow \mathtt{robot}_1\}$$

Now consider a new text at test time "*throw away the chips*", which does not explicitly mention where to throw the chips. Our semantic parsing algorithm (Section 5) will use the previous mapping $y \rightarrow \mathtt{garbabeBin}_1$ to choose an object most similar to a garbage bin.

5 Semantic Parsing Model

Given a sequence of frame nodes $c_{1:k}$ and an initial environment e_1, our semantic parsing model defines a joint distribution over logical forms $z_{1:k}$. Specifically, we define a conditional random field (CRF) over $z_{1:k}$, as shown in Figure 2:

$$p_\theta(z_{1:k} \mid c_{1:k}, e_1) \propto \exp\left(\sum_{i=1}^{k} \phi(c_i, z_{i-1}, z_i, e_i) \cdot \theta\right), \quad (1)$$

where $\phi(c_i, z_{i-1}, z_i, e_i)$ is the feature vector and θ is the weight vector. Note that the environments $e_{1:k}$ are a deterministic function of the logical forms $z_{1:k}$ through the recurrence $e_{i+1} = \mathit{simulator}(e_i, \mathit{planner}(e_i, z_i))$, which couples the different time steps.

[1] We use the symbolic planner of Rintanen (2012) which can perform complex planning. For example, to pick up a bottle that is blocked by a stack of books, the planner will first remove the books before grasping the bottle. In contrast, Artzi and Zettlemoyer (2013) use a simple search over implicit actions.

Features. The feature vector $\phi(c_i, z_{i-1}, z_i, e_i)$ contains 16 features which capture the dependencies between text, logical forms, and environment. Recall that $z_i = ([\nu \Rightarrow (\lambda\vec{v}.S, \xi)], \xi_i)$, where ξ is the environment in which the lexical entry was created and ξ_i is the current environment. Let $f_i = (\lambda\vec{v}.S)(\xi_i)$ be the current postcondition. Here we briefly describe the important features (see the supplemental material for the full list):

- *Language and logical form*: The logical form z_i should generally reference objects mentioned in the text. Assume we have computed a correlation $\rho(\omega, o)$ between each object description ω and object o, whose construction is described later. We then define two features: precision correlation, which encourages z_i to only use objects referred to in c_i; and recall correlation, which encourages z_i to use all the objects referred to in c_i.
- *Logical form*: The postcondition f_i should be based on previously seen environments. For example, microwaving an empty cup and grasping a couch are unlikely postconditions. We define features corresponding to the average probability (based on the training data) of all conjunctions of at most two atoms in the postcondition (e.g., $\texttt{grasping}(\texttt{robot}, \texttt{cup})\}$). We do the same with their abstract versions ($\{\texttt{grasping}(\texttt{v}_1, \texttt{v}_2)\}$). In addition, we build the same set of four probability tables conditioned on verbs in the training data. For example, the abstract postcondition $\texttt{state}(\texttt{v}_1, \texttt{water})$ has a higher probability conditioned on the verb "*fill*". This gives us a total of 8 features of this type.
- *Logical form and environment*: Recall that anchoring helps us in dealing with ellipsis and noise. We add a feature based on the average correlation between the objects of the new mapping ξ_i with the corresponding objects in the anchored mapping ξ.

The other features are based on the relationship between object descriptions, similarity between ξ and ξ_i and transition probabilities between logical forms z_{i-1} and z_i. These probabilities are also learned from training data.

Mapping Object Descriptions. Our features rely on a mapping from object descriptions ω (e.g., "*the red shiny cup*") to objects o (e.g., $\texttt{cup}_8$), which has been addressed in many recent works (Matuszek et al., 2012a; Guadarrama et al., 2014; Fasola and Matari'c, 2014).

One key idea is: instead of computing rigid lexical entries such as *cup* $\rightarrow$ $\texttt{cup}_1$, we use a continuous correlation score $\rho(\omega, o) \in [0, 1]$ that measures how well ω describes o. This flexibility allows the algorithm to use objects not explicitly mentioned in text. Given "*get me a tank of water*", we might choose an approximate vessel (e.g., $\texttt{cup}_2$).

Given an object description ω, an object o, and a set of previously seen objects (used for anaphoric resolution), we define the correlation $\rho(\omega, o)$ using the following approach:

- If ω *is a pronoun*, $\rho(\omega, o)$ is the ratio of the position of the last reference of o to the length of the action sequence computed so far, thus preferring recent objects.
- Otherwise, we compute the correlation using various sources: the object's category; the object's state for handling metonymy (e.g., the description "*coffee*" correlates well with the object $\texttt{mug}_1$ if $\texttt{mug}_1$ contains coffee—$\texttt{state}(\texttt{mug}_1, \texttt{has-coffee})$ is true), WordNet (Fellbaum, 1998) for dealing synonymy and hyponymy; and word alignments between the objects and text from Giza++ (Och and Ney, 2003) to learn domain-specific references (e.g., "*Guinness book*" refers to $\texttt{book}_1$, not $\texttt{book}_2$). More details can be found in the supplemental material.

6 Lexicon Induction from Training Data

In order to map text to logical forms, we first induce an initial anchored lexicon Λ from the training data $\{(x^{(m)}, e^{(m)}, a^{(m)}, \pi^{(m)})\}_{m=1}^{M}$. At test time, we add new lexical entries (Section 7) to Λ.

Recall that shallow parsing $x^{(m)}$ yields a list of frame nodes $c_{1:k}$. For each frame node c_i and its aligned action sequence a_i, we take the conjunction of all the atoms (and their negations) which are false in the current one e_i but true in the next environment e_{i+1}. We parametrize this conjunction by replacing each object with a variable, yielding a postcondition S parametrized by free variables $\vec{v}$ and the mapping ξ from $\vec{v}$ to objects in e_i. We then add the lexical entry $[verb(c_i) \Rightarrow (\lambda\vec{v}.S, \xi)]$ to Λ.

Instantiating Lexical Entries. At test time, for a given clause c_i and environment e_i, we generate set of logical forms $z_i = (\ell_i, \xi_i)$. To do this, we consider the lexical entries in Λ with the same verb as c_i. For each such lexical entry ℓ_i, we can map its free variables $\vec{v}$ to objects in e_i in an exponential number of ways. Therefore, for each ℓ_i we only consider the logical form (ℓ_i, ξ_i) where the mapping ξ_i obtains the highest score under the current model: $\xi_i = \arg\max_{\xi'} \phi(c_i, z_{i-1}, (\ell_i, \xi'), e_i) \cdot \theta$. For the feature vector ϕ that we consider, this approximately translates to solving an integer quadratic program with variables $[y_{ij}] \in \{0, 1\}$, where $y_{ij} = 1$ only if v_i maps to object j.

7 Environment-Driven Lexicon Induction at Test Time

Unfortunately, we cannot expect the initial lexicon Λ induced from the training set to have full coverage of the required postconditions. Even after using 90% of the data for training, we encountered 17% new postconditions on the remaining 10%. We therefore propose generating new lexical entries at test time and adding them to Λ.

Formally, for a given environment e_i and frame node c_i, we want to generate likely logical forms. Although the space of all possible logical forms is very large, the environment constrains the possible interpretations. We first compute the set of atoms that are false in e_i and that only contain objects o that are "referred" to by either c_i or c_{i-1}, where "refers" means that there exists some argument ω in c_i for which $o \in \arg\max_{o'} \rho(\omega, o')$. For example, if c_i corresponds to the text "*distribute pillows among the couches*", we consider the atom $\text{on}(\texttt{pillow}_1, \texttt{armchair}_1)$ but not $\text{on}(\texttt{pillow}_1, \texttt{snacktable}_2)$ since the object $\texttt{armchair}_1$ has the highest correlation to the description "*couches*".

Next, for each atom, we convert it into a logical form $z = (\ell, \xi)$ by replacing each object with a variable. While this generalization gives us a mapping ξ, we create a lexical entry $\ell_i = [\nu \Rightarrow (\lambda\vec{v}.S, \emptyset)]$ without it, where S is the parameterized atom. Note that the anchored mapping is empty, representing the fact that this lexical entry was unseen during training time. For example, the atom $\texttt{state}(\texttt{tv}_1, \texttt{mute})$ would be converted to the logical form (ℓ, ξ), where $\ell = [verb(c_i) \Rightarrow (\lambda v.\texttt{state}(v, \texttt{mute}), \emptyset]$ and $\xi = \{v \rightarrow \texttt{tv}_1\}$. We do not generalize state names (e.g., $\texttt{mute}$) because they generally are part of the meaning of the verb.

The score $\phi(c_i, z_{i-1}, z_i, e_i) \cdot \theta$ is computed for the logical form z_i produced by each postcondition. We then take the conjunction of every pair of postconditions corresponding to the 200 highest-scoring logical forms. This gives us new set of postconditions on which we repeat the generalization-scoring-conjunction cycle. We keep doing this while the scores of the new logical forms is increasing or while there are logical forms remaining.

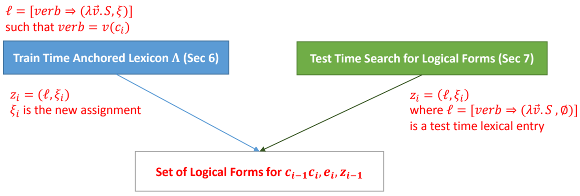
$\ell = [verb \Rightarrow (\lambda\vec{v}.S, \xi)]$
such that $verb = v(c_i)$
Train Time Anchored Lexicon Λ (Sec 6)
Test Time Search for Logical Forms (Sec 7)
$z_i = (\ell, \xi_i)$
ξ_i is the new assignment
$z_i = (\ell, \xi_i)$
where $\ell = [verb \Rightarrow (\lambda\vec{v}.S, \emptyset)]$
is a test time lexical entry
Set of Logical Forms for $c_{i-1}c_i, e_i, z_{i-1}$

Figure 4: Logical forms for a given clause c_i, environment e_i, and previous logical form z_{i-1} are generated from both a lexicon induced from training data and a test-time search procedure based on the environment.

If a logical form $z = ([\nu \Rightarrow (\lambda\vec{v}.S, \emptyset)], \xi)$ is used by the predicted action sequence, we add the lexical entry $[\nu \Rightarrow (\lambda\vec{v}.S, \xi)]$ to the lexicon Λ. This is different to other lexicon induction procedures such as GENLEX (Zettlemoyer and Collins, 2007) which are done at training time only and require more supervision. Moreover, GENLEX does not use the environment context in creating new lexical entries and thus is not appropriate at test time, since it would vastly overgenerate lexical entries compared to our approach. For us, the environment thus provides implicit supervision for lexicon induction.

8 Inference and Parameter Estimation

Inference. Given a text x (which is converted to $c_{1:k}$ via Section 3.2) and an initial environment e_1, we wish to predict an action sequence a based on $p_\theta(a_{1:k} \mid c_{1:k}, e_1)$, which marginalizes over all logical forms $z_{1:k}$ (see Figure 2).

To enumerate possible logical forms, semantic parsers typically lean heavily on a lexicon (Artzi and Zettlemoyer, 2013), leading to high precision but lower recall, or search more aggressively (Berant et al., 2013), leading to higher recall but lower precision. We adopt the following hybrid approach: Given e_i, c_{i-1}, c_i and z_{i-1}, we use both the lexical entries in Λ as explained in Section 6 and the search procedure in Section 7 to generate the set of possible logical forms for z_i (see Figure 4). We use beam search, keeping only the highest-scoring logical form with satisfiable postconditions for each $i \in \{1, \ldots, k\}$ and resulting action sequence $a_{1:i}$.

Parameter Estimation. We split 10% of our training data into a separate tuning set (the 90% was used to infer the lexicon). On each example in this set, we extracted the full sequence of logical forms $z_{1:k}$ from the action sequence $a_{1:k}$ based on Section 6. For efficiency, we used an objective similar to pseudolikelihood to estimate the parameters θ. Specifically, we maximize the average log-likelihood over each adjacent pair of logical forms under $\tilde{p}_\theta$:

$$\tilde{p}_\theta(z_i \mid z_{i-1}, c_i, e_i) \propto \exp(\phi(c_i, z_{i-1}, z_i, e_i)^\top \theta). \quad (2)$$

The weights were initialized to 0. We performed 300 iterations over the validation set with a learning rate of $\frac{0.005}{N}$.

9 Dataset and Experiments

9.1 Dataset

We collected a dataset of 500 examples from 62 people using a crowdsourcing system similar to Misra et al. (2014). We consider two different 3D scenarios: a kitchen and a living room, each containing an average of 40 objects. Both of these scenarios have 10 environments consisting of different sets of objects in different configurations. We define 10 high-level objectives, 5 per scenario, such as *clean the room*, *make coffee*, *prepare room for movie night*, etc.

One group of users wrote natural language commands to achieve the high-level objectives. Another group controlled a virtual robot to accomplish the commands given by the first group. The dataset contains considerable variety, consisting of 148 different verbs, an average of 48.7 words per text, and an average of 21.5 actions per action sequence. Users make spelling and grammar errors in addition to occasionally taking random actions not relevant to the text. The supplementary material contains more details.

We filtered out 31 examples containing fewer than two action sequences. Of the remaining examples, 378 were used for training and 91 were used for test. Our algorithm is tested on four new environments (two from each scenario).

9.2 Experiments and Results

Evaluation Metrics. We consider two metrics, *IED* and *END*, which measure accuracy based on the action sequence and environment, respectively. Specifically, the *IED* metric (Misra et al., 2014) is the edit distance between predicted and true action sequence. The *END* metric is the Jaccard index of sets A and B, where A is the set of atoms (e.g., $\texttt{on(cup}_1\texttt{,table}_1\texttt{))}$ whose truth value changed due to simulating the predicted action sequence, and B is that of the true action sequence.

Baselines. We compare our algorithm with the following baselines:

Table 3: Results on the metrics and baselines described in section 9.2. The numbers are normalized to 100 with larger values being better.

Algorithm	IED	END
Chance	0.3	0.5
Manually Defined Templates	2.5	1.8
UBL- Best Parse (Kwiatkowski et al., 2010)	5.3	6.9
VEIL (Misra et al., 2014)	14.8	20.7
Model with only train-time lexicon induction	20.8	26.8
Model with only test-time lexicon induction	21.9	25.9
Full Model	**22.3**	**28.8**

1. Chance: Randomly selects a logical form for every frame node from the set of logical forms generated by generalizing all possible postconditions that do not hold in the current environment. These postconditions could contain up to 93 atoms.

2. Manually Defined Templates: Defines a set of postcondition templates for verbs similar to Guadarrama (2013).

3. UBL-Best Parse (Kwiatkowski et al., 2010): UBL algorithm trained on text aligned with postconditions and a noun-phrase seed lexicon. The planner uses the highest scoring postcondition given by UBL to infer the action sequence.

4. VEIL (Misra et al., 2014): Uses action sequences as logical forms and does not generate lexical entries at test time.

We also consider two variations of our model: (i) using only lexical entries induced using the training data, and (ii) using only the logical forms induced at test-time by the search procedure.

The results are presented in Table 3. We observe that our full model outperforms the baseline and the two pure search- and lexicon-based variations of our model. We further observe that adding the search procedure (Section 7) improved the accuracy by 1.5% on IED and 2% on END. The logical forms generated by the search were able to successfully map 48% of the new verbs.

Table 2 shows new verbs and concepts that the algorithm was able to induce at test time. The algorithm was able to correctly learn the lexical entries for the verbs "*distribute*" and "*mix*", while the ones for verbs "*change*" and "*boil*" were only partly correct. The postconditions in Table 2 are not structurally isomorphic to previously-seen logical forms; hence they could not have been handled by using synonyms or factored lexicons (Kwiatkowski et al., 2011). The poor performance of UBL was because the best logical form often produced an unsatisfiable postcondition. This can be remedied by joint modeling with the environ-

Table 2: New verbs and concepts induced at test time (Section 7).

Text	Postcondition represented by the learned logical form	# Log. forms explored
"mix it with ice cream and syrup"	$\texttt{state}(\texttt{cup}_2, \texttt{ice-cream}_1) \wedge \texttt{state}(\texttt{cup}_2, \texttt{vanilla})$	15
"distribute among the couches"	$\wedge_{\mathtt{j} \in \{1,3\}} \texttt{on}(\texttt{pillow}_\mathtt{j}, \texttt{loveseat}_1) \wedge \texttt{on}(\texttt{pillow}_{\mathtt{i}+1}, \texttt{armchair}_{\mathtt{i}+1})$	386
"boil it on the stove"	$\texttt{state}(\texttt{stove}, \texttt{stovefire1}) \wedge \texttt{state}(\texttt{kettle}, \texttt{water})$	109
"change the channel to a movie"	$\texttt{state}(\texttt{tv}_1, \texttt{channel4}) \wedge \texttt{on}(\texttt{book}_1, \texttt{loveseat}_1)$	98

ment. The VEIL baseline used actions for representation and does not generalize as well as the postconditions in our logical forms.

It is also instructive to examine the alternate postconditions that the search procedure considers. For the first example in Table 2, the following postcondition was considered by not selected:

$\texttt{grasping}(\texttt{robot}, \texttt{icecream}_2) \wedge \texttt{grasping}(\texttt{robot}, \texttt{syrup}_1)$

While this postcondition uses all the objects described in the text, the environment-based features suggest it makes little sense for the task to end with the robot eternally grasping objects. For the second example, alternate postconditions considered included:

1. $\texttt{on}(\texttt{pillow}_1, \texttt{pillow}_2) \wedge \texttt{on}(\texttt{pillow}_3, \texttt{pillow}_4)$
2. $\wedge_{\mathtt{j}=1}^{4} \texttt{on}(\texttt{pillow}_\mathtt{j}, \texttt{loveseat}_1)$
3. $\wedge_{\mathtt{j}=1}^{3} \texttt{near}(\texttt{robot}_1, \texttt{armchair}_\mathtt{j})$

The algorithm did not choose options 1 or 3 since the environment-based features recognizes these as unlikely configurations. Option 2 was ruled out since the recall correlation feature realizes that not all the couches are mentioned in the postcondition.

To test how much features on the environment help, we removed all such features from our full model. We found that the accuracy fell to 16.0% on the IED metric and 16.6% on the END metric, showing that the environment is crucial.

In this work, we relied on a simple deterministic shallow parsing step. We found that shallow parsing was able to correctly process the text in only 46% of the test examples, suggesting that improving this initial component or at least modeling the uncertainty there would be beneficial.

10 Related Work

Our work uses semantic parsing to map natural language instructions to actions via novel concepts, which brings together several themes: actions, semantic parsing, novel concepts, and robotics.

Mapping Text to Actions. Several works (Branavan et al., 2009; Branavan et al., 2010; Vogel and Jurafsky, 2010) use reinforcement learning to directly map to text to actions, and do not even require an explicit model of the environment. However, they can only handle simple actions, whereas our planner and simulator allows us to work with postconditions, and thus tackle high-level instructions. Branavan et al. (2012) extract precondition relations from text, learn to map text to subgoals (postconditions) for a planner. However, their postconditions are atomic, whereas ours are complex conjunctions.

Other works (Chen and Mooney, 2011; Kim and Mooney, 2012; Kollar et al., 2010; Fasola and Mataric, 2013) have focused only on navigational verbs and spatial relations, but do not handle high-level verbs. Artzi and Zettlemoyer (2013) also fall into the above category and offer a more compositional treatment. They focus on how words compose; we focus on unraveling single words.

The broader problem of grounded language acquisition, involving connecting words to aspects of a situated context has been heavily studied (Duvallet et al., 2014; Yu and Siskind, 2013; Chu et al., 2013; Chen and Mooney, 2008; Mooney, 2008; Fleischman and Roy, 2005; Liang et al., 2009).

Semantic Parsing. In semantic parsing, much work has leveraged CCG (Zettlemoyer and Collins, 2005; Zettlemoyer and Collins, 2007; Kwiatkowski et al., 2010). One challenge behind lexically-heavy approaches is ensuring adequate lexical coverage. Kwiatkowski et al. (2011) enhanced generalization by factoring a lexical entry into a template plus a lexeme, but the rigidity of the template remains. This is satisfactory when words map to one (or two) predicates, which is the case in most existing semantic parsing tasks. For example, in Artzi and Zettlemoyer (2013), verbs are associated with single predicates ("*move*" to move, "*walk*" to walk, etc.) In our setting, verbs contain multi-predicate postconditions, for which these techniques would not be suitable.

As annotated logical forms for training semantic parsers are expensive to obtain, several works (Clarke et al., 2010; Liang et al., 2011; Berant et al., 2013; Kwiatkowski et al., 2013) have developed methods to learn from weaker supervision, and as in our work, use the execution of the logical forms to guide the search. Our supervision is even weaker in that we are able to learn at test time from partial environment constraints.

Grounding to Novel Concepts. Guadarrama et al. (2014) map open vocabulary text to objects in an image using a large database. Matuszek et al. (2012a) create new predicates for every new adjective at test time. Others (Kirk et al., 2014) ask users for clarification. In contrast, we neither have access to large databases for this problem, nor do we do create new predicates or use explicit supervision at test time.

Robotic Applications. Our motivation behind this work is to build robotic systems capable of taking commands from users. Other works in this area have considered mapping text to a variety of manipulation actions (Sung et al., 2015). Levine et al. (2015) and Lenz et al. (2015) focus on specific manipulation actions. In order to build a representation of the environment, Ren et al. (2012) and Wu et al. (2014) present vision algorithms but only output symbolic labels, which could act as inputs to our system. In future work, we also plan to integrate our work with RoboBrain (Saxena et al., 2014) to leverage these existing systems for building a robotic system capable of working with physical world data.

11 Conclusion

We have presented an algorithm for mapping text to actions that induces lexical entries at test time using the environment. Our algorithm couples the lexicon extracted from training data with a test-time search that uses the environment to reduce the space of logical forms. Our results suggest that using the environment to provide lexical coverage of high-level concepts is a promising avenue for further research.

Acknowledgements. This research was supported by the ONR (award N00014-14-1-0156), a Sloan Research Fellowship to the third author, and a Microsoft Research Faculty Fellowship and NSF Career Award to the fourth author. We thank Aditya Jami and Jaeyong Sung for useful discussions. We also thank Jiaqi Su for her help with data collection and all the people who participated in the user study.

Reproducibility. Code, data, and experiments for this paper are available on the CodaLab platform at `https://www.codalab.org/worksheets/0x7f9151ec074f4f589e4d4786db7bb6de/`. Demos can be found at `http://tellmedave.com`.

Appendix: Parsing Text into Control Flow Graph.

We first decompose the text x into its control flow graph G using a simple set of rules:

- The parse tree of x is generated using the Stanford parser (Klein and Manning, 2003) and a frame node is created for each non-auxiliary verb node in the tree.
- Conditional nodes are discovered by looking for the keywords *until, if, after, when.* The associated subtree is then parsed deterministically using a set of a rules. For example, a rule parses *"for x minutes"* to `for(digit:x,unit:minutes)`. We found that all conditionals can be interpreted against the initial environment e_1, since our world is fully-observable, deterministic, and the user giving the command has full view of the world.
- To find objects, we look for anaphoric terminal nodes or nominals whose parent is not a nominal or which have a PP sibling. These are processed into object descriptions ω.
- Object descriptions ω are attached to the frame node, whose verb is nearest in the parse tree to the main noun of ω.
- Nodes corresponding to $\{$*IN,TO,CC,","*$\}$ are added as the relation between the corresponding argument objects.
- If there is a conjunction between two objects in a frame node and if these objects have the same relation to other objects, then we split the frame node into two sequential frame nodes around these objects. For example, a frame node corresponding to the text segment "*take the cup and bowl from table*" is split into two frame nodes corresponding to "*take the cup from table*" and "*take bowl from table*".
- A temporal edge is added between successive frame nodes in the same branch of a condition. A temporal edge is added between a conditional node and head of the true and false branches of the condition. The end of all branches in a sentence are joined to the starting node of the successive sentence.

References

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

J. Berant and P. Liang. 2014. Semantic parsing via paraphrasing. In *Association for Computational Linguistics (ACL)*.

J. Berant, A. Chou, R. Frostig, and P. Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Empirical Methods in Natural Language Processing (EMNLP)*.

M. Bollini, J. Barry, and D. Rus. 2011. Bakebot: Baking cookies with the PR2. In *The PR2 Workshop, IROS*.

S. Branavan, H. Chen, L. S. Zettlemoyer, and R. Barzilay. 2009. Reinforcement learning for mapping instructions to actions. In *Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 82–90.

S. Branavan, L. Zettlemoyer, and R. Barzilay. 2010. Reading between the lines: Learning to map high-level instructions to commands. In *Association for Computational Linguistics (ACL)*, pages 1268–1277.

S. Branavan, N. Kushman, T. Lei, and R. Barzilay. 2012. Learning high-level planning from text. In *Association for Computational Linguistics (ACL)*, pages 126–135.

D. L. Chen and R. J. Mooney. 2008. Learning to sportscast: A test of grounded language acquisition. In *International Conference on Machine Learning (ICML)*, pages 128–135.

D. L. Chen and R. J. Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *Association for the Advancement of Artificial Intelligence (AAAI)*, pages 859–865.

V. Chu, I. McMahon, L. Riano, C. McDonald, Q. He, J. Perez-Tejada, M. Arrigo, N. Fitter, J. Nappo, T. Darrell, et al. 2013. Using robotic exploratory procedures to learn the meaning of haptic adjectives. In *International Conference on Intelligent Robots and Systems (IROS)*.

J. Clarke, D. Goldwasser, M. Chang, and D. Roth. 2010. Driving semantic parsing from the world's response. In *Computational Natural Language Learning (CoNLL)*, pages 18–27.

F. Duvallet, M. R. Walter, T. Howard, S. Hemachandra, J. Oh, S. Teller, N. Roy, and A. Stentz. 2014. Inferring maps and behaviors from natural language instructions. In *International Symposium on Experimental Robotics (ISER)*.

J. Fasola and M. Mataric. 2013. Using semantic fields to model dynamic spatial relations in a robot architecture for natural language instruction of service robots. In *International Conference on Intelligent Robots and Systems (IROS)*.

J. Fasola and M. J. Matari'c. 2014. Interpreting instruction sequences in spatial language discourse with pragmatics towards natural human-robot interaction. In *International Conference on Robotics and Automation (ICRA)*, pages 6667–6672.

C. Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

M. Fleischman and D. Roy. 2005. Intentional context in situated natural language learning. In *Computational Natural Language Learning (CoNLL)*, pages 104–111.

S. Guadarrama, L. Riano, D. Golland, D. Gouhring, Y. Jia, D. Klein, P. Abbeel, and T. Darrell. 2013. Grounding spatial relations for human-robot interaction. In *International Conference on Intelligent Robots and Systems (IROS)*.

S. Guadarrama, E. Rodner, K. Saenko, N. Zhang, R. Farrell, J. Donahue, and T. Darrell. 2014. Open-vocabulary object retrieval. In *Robotics: Science and Systems (RSS)*.

J. Kim and R. Mooney. 2012. Unsupervised PCFG induction for grounded language learning with highly ambiguous supervision. In *Computational Natural Language Learning (CoNLL)*, pages 433–444.

N. H. Kirk, D. Nyga, and M. Beetz. 2014. Controlled natural languages for language generation in artificial cognition. In *International Conference on Robotics and Automation (ICRA)*, pages 6667–6672.

D. Klein and C. Manning. 2003. Accurate unlexicalized parsing. In *Association for Computational Linguistics (ACL)*, pages 423–430.

T. Kollar, S. Tellex, D. Roy, and N. Roy. 2010. Grounding verbs of motion in natural language commands to robots. In *International Symposium on Experimental Robotics (ISER)*.

T. Kwiatkowski, L. Zettlemoyer, S. Goldwater, and M. Steedman. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1223–1233.

T. Kwiatkowski, L. Zettlemoyer, S. Goldwater, and M. Steedman. 2011. Lexical generalization in CCG grammar induction for semantic parsing. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1512–1523.

T. Kwiatkowski, E. Choi, Y. Artzi, and L. Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Empirical Methods in Natural Language Processing (EMNLP)*.

I. Lenz, R. Knepper, and A. Saxena. 2015. Deepmpc: Learning deep latent features for model predictive control. In *Robotics Science and Systems (RSS)*.

S. Levine, C. Finn, T. Darrell, and P. Abbeel. 2015. End-to-end training of deep visuomotor policies. *arXiv preprint arXiv:1504.00702*.

P. Liang, M. I. Jordan, and D. Klein. 2009. Learning semantic correspondences with less supervision. In *Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 91–99.

P. Liang, M. I. Jordan, and D. Klein. 2011. Learning dependency-based compositional semantics. In *Association for Computational Linguistics (ACL)*, pages 590–599.

C. Matuszek, N. FitzGerald, L. Zettlemoyer, L. Bo, and D. Fox. 2012a. A joint model of language and perception for grounded attribute learning. In *International Conference on Machine Learning (ICML)*, pages 1671–1678.

C. Matuszek, E. Herbst, L. Zettlemoyer, and D. Fox. 2012b. Learning to parse natural language commands to a robot control system. In *International Symposium on Experimental Robotics (ISER)*.

D. Misra, J. Sung, K. Lee, and A. Saxena. 2014. Tell Me Dave: Context-sensitive grounding of natural language to mobile manipulation instructions. In *Robotics: Science and Systems (RSS)*.

R. Mooney. 2008. Learning to connect language and perception. In *Association for the Advancement of Artificial Intelligence (AAAI)*, pages 1598–1601.

F. J. Och and H. Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29:19–51.

X. Ren, L. Bo, and D. Fox. 2012. Rgb-(d) scene labeling: Features and algorithms. In *Computer Vision and Pattern Recognition (CVPR)*, pages 2759–2766.

J. Rintanen. 2012. Planning as satisfiability: Heuristics. *Artificial Intelligence*, 193.

A. Saxena, A. Jain, O. Sener, A. Jami, D. K. Misra, and H. S. Koppula. 2014. Robobrain: Large-scale knowledge engine for robots. *arXiv preprint arXiv:1412.0691*.

J. Sung, S. H. Jin, and A. Saxena. 2015. Robobarista: Object part based transfer of manipulation trajectories from crowd-sourcing in 3d pointclouds. *arXiv preprint arXiv:1504.03071*.

S. Tellex, T. Kollar, S. Dickerson, M. R. Walter, A. G. Banerjee, S. J. Teller, and N. Roy. 2011. Understanding natural language commands for robotic navigation and mobile manipulation. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

A. Vogel and D. Jurafsky. 2010. Learning to follow navigational directions. In *Association for Computational Linguistics (ACL)*, pages 806–814.

C. Wu, I. Lenz, and A. Saxena. 2014. Hierarchical semantic labeling for task-relevant RGB-D perception. In *Robotics: Science and Systems (RSS)*.

H. Yu and J. M. Siskind. 2013. Grounded language learning from video described with sentences. In *Association for Computational Linguistics (ACL)*, pages 53–63.

L. S. Zettlemoyer and M. Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Uncertainty in Artificial Intelligence (UAI)*, pages 658–666.

L. S. Zettlemoyer and M. Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL)*, pages 678–687.