Supplementary Material for the Paper: 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

1 Dataset: Samples and Challenges

As described in the main paper, we collected a dataset $D = (x^{(n)}, e^{(n)}, a^{(n)}, \pi^{(n)})_{n=1}^{500}$.

Environment Complexity. Our environments are 3D scenarios consisting of complex objects such as fridge, microwave and television with many states. These objects can be in different spatial relations with respect to other objects. For example, *"bag of chips"* can be found behind the television. Figure 1 shows some sample environments from our dataset. For example, an object of category television consists of 6 channels, volume level and power status. An object can have different values of states in different environment and different environment consists of different set of objects and their placement. For example, television might be powered on in one environment and closed in another, microwave might have an object inside it or not in different environment, etc.

Moreover, there are often more than one object of the same category. For example, our environment typically have two books, five couches, four pillows etc. Objects of the same category can have different appearance. For example, a book can have the cover of a math book or a Guinness book of world record; resulting in complex object descriptions such as in *"throw the sticky stuff in the bowl"*. They can also have the same appearance making people use spatial relations or other means while describing them such as in *"get me the cup next to the microwave"*. This dataset is significantly more challenging compared to the 2D navigation dataset or GUI actions in windows dataset considered earlier.

Task Complexity. In this paper, we consider tasks with high level objective such as *clean the room, prepare the room for movie night* etc. compared to navigation or simple manipulations tasks involving picking and placing objects. This results in extremely free-form text such as shown below:

- *"Turn on xbox. Take Far Cry Game CD and put in xbox by pressig eject to open drive. Throw out beer, coke, and sketchy stuff in bowl. Take pillows from shelf and distribute among couches ."*
- *"Boil some water and make some coffee. Find a white bowl. Take ice cream out of the freezer. Put coffee into the white bowl, then put two scoops of ice cream over that . Finally, take the syrup on the counter and drizzle it over the ice cream."*
- *"If anything is disposable and used, put it in the trash bag. If it is not disposable and on the floor, put it on the table nearest any items associated with it. If it is not disposable and on the floor, put it on the table nearest any items associated with it. If it is not disposable and on the floor, put it on the table nearest any items associated with it. If a not disposable item contains only disposable objects, dump them into the trash bag, and treat the object like it was on the floor. Remove the trash bag from the scenario."*
- *"Make some coffee. Make some eggs on the stove and then put them on a plate and serve the eggs and coffee to me "*
- *"Take Book of Records and place on table with brown book. The TV is already turned off .Throw out open beer and coke. Chips are good."*
- *"Dump the coffee in the mug in the sink, put all perishable items in the refrigerator, put all the dishes, utensils, and pots in the sink."*
- *"Turn TV on with remote and find movie (Lincoln) on with remote and find movie (Lincoln) . Take bag of chips and place on table. Take pillow from shelf and place on a sofa. Throw away beer and soda, and place Book of Records on shelf with brown book."*

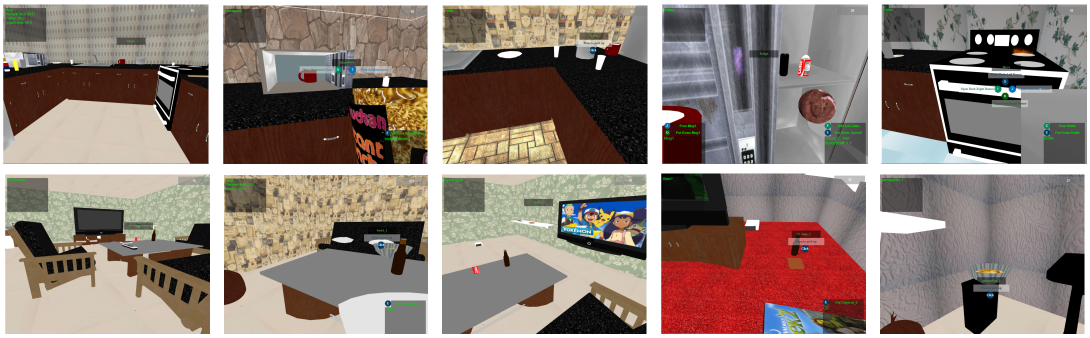

Figure 1: Sample of 3D Environments that we consider. Environments consists of several objects, each object can have several states. Different environment have different set of objects with different configuration. There can be more than one objects of the same category.

- *"Mix syrup and water to make a drink. You can get water by rotating the tab near sink. Use kettle to boil water and mix heated water with instantRamen."*

we refer the readers to the full dataset for more examples.

Noise in the dataset. Our dataset was collected from non-expert users including the action sequences. Therefore, our dataset had considerable noise as is also visible from the examples above. The noise included spelling and grammar errors in the text, text that is asking the robot to do things which it cannot do such as moving the chairs, noise in the action sequences and noise in aligning parts of action sequences and the text segments.

We use a set of rules to remove noise from the dataset, such as removing cyclic patterns in the action sequence. This often happened when users tried to give a demonstration to the robot such as keeping a mug inside the microwave, but made an error and hence repeated the actions. We want to emphasize here that, the average length of 21.5 actions for the action sequences in the dataset was derived after removing this noise.

Out of the 500 points that we collected, we further removed 31 points consisting of action sequences of length less than 2.

2 Examples of Planning and Simulation

We use a planner and a simulator which allows us to use post-conditions in defining our logical forms. In order to perform planning and simulation— we encode the domain knowledge in STRIPS planning format. This defines preconditions and effect of action on the environment. An example is given below:

(:action `release` *:parameters* (*o*)
:precondition (`grasping robot o`)
:effect (`not` (`grasping robot o`)))

This STRIPS program defines the action `release` which takes an object o as the argument. The precondition of this action is that the robot must be grasping the object o and the effect is that robot releases the object o.

3 Mapping Object Descriptions

Given an object description ω and a set of physical objects $\{o_j\}_{j=1}^{m}$; we want to find the correlation $\rho(\omega, o_j) \in [0, 1]$ of how well does the description ω describes the object o_j. When the description is not a pronoun, we take the following approach. We initialize $\forall_j\ \rho(\omega, o_j) = 0$ and then try the following rules in the given order, stopping after the first match:

- *category matching*: if there exists a set of objects $\{o'_j\}$ containing part of the description in its name then we define $\forall_j \rho(\omega, o'_j) = 1$.
- *containment (metonymy)*: for every object o_j; if the main noun in ω matches the state-name of a state of o_j which has value $True$ then we define $\rho(\omega, o_j) = 1$.
- *wordnet similarity*: for every object o_j we find $\rho(\omega, o_j)$ using a modified Lesk algorithm based on WordNet. If a similarity score greater than 0.85 is found then we return.
- *domain specific references*: We use giza-pp algorithm to learn translation probabilities between text and corresponding action sequences, using the training data. This gives us a probability table T[*words,object-name*] of words in text and object name in the sequence. We then initialize $\rho(\omega, o_j)$ by averaging the value of $T[w, o_j.name]$ for every word w in ω.

4 Manual Rules for Parsing Conditions

As explained in the paper, we parse conditional expressions into their meaning representations using a set of rules. This was possible and motivated both by the fact that the conditional expressions in our dataset are easy and because meaning representations of conditional expressions are not observed in the dataset (which only contains actions corresponding to frame nodes). We parse conditional expressions using the following deterministic rules.

string which is a noun or a pronoun → *object*
string representing a state-name → *statename*
string representing a spatial relation → *relation*
"minute"|"min"|"hour"|"sec"|"seconds" → time-unit
object statename → *state(object, statename)*
string1 relation string2 → *relation(string1, string2)*
digit time-unit → *time(digit, time-unit)*
for/when/after/until *state(object, statename)* →
for/when/after/until(state(object, statename))
for/after/until *time(digit,unit)* →
for/after/until(time(digit,unit))
if *state(string1, string2)* → *if(state(string1, string2))*

Each word in the text can further be ignored i.e. mapped to ϵ. These rules are simple enough to be parsed in a bottom up fashion starting with words. *Example: "for 3 minutes"* is parsed as:
minutes→ time-unit:min
3 time-unit:min→ time(digit:3,time-unit:min)
for time(3,time-unit:min)→ for(time(digit:3,time-unit:min))

For *"if"* condition, which has a true and false branch; we evaluate the condition using the starting environment. In case of a parsing failure, we always return true.

5 Feature Equation

We use the following features $\phi(c_i, z_{i-1}, z_i, e_i)$ briefly described in the paper. The logical form is given by $z_i = ([\nu \Rightarrow (\lambda\vec{v}.S, \xi)], \xi_i)$. Here ξ_i, ξ are mappings of the variable $\vec{v}$ of the parametrized post-condition S. Let $\vec{v}$ have m variables and $\xi(v_j)$ represents the object in e_i, to which the variable v_j is mapped using ξ. Further the post-condition f_i is given by $f_i = (\lambda\vec{v}.S)(\xi_i)$:

- *Language and Logical Form*: There are two features of this type:

$$f_{le} = \frac{1}{m}\sum_{j=1}^{m} \max_{\omega} \rho(\omega, \xi_i(v_j))$$

$$f_{recall} = \frac{1}{|\omega \in c_i|}\sum_{\omega \in c_i} \max_{j} \rho(\omega, \xi_i(v_j))$$

where ρ is the object description correlation score(see paper). For the f_{LE} feature, we also consider the previous clause c_{i-1} in the computation of $\max_\omega \rho(\omega, \xi_i(v_j))$.

- *Logical Form*: We prefer the post-conditions which have high environment priors and are therefore likely to occur again. Let post-condition $f_i = \wedge_l f_{il} = f_{i1} \wedge f_{i2} \cdots f_{ip}$ consists of p atomic-predicates (or their negations) given by f_{il}. Also let, $pm(\wedge_l f_{il})$ be the parametrized version of the post-condition $\wedge_l f_{il}$ created by replacing each unique object by a unique variable. Example, the post-condition $on(cup_2, bowl_3) \wedge state(cup_2, water)$ is parametrized to $on(v_1, v_2) \wedge state(v_1, water)$. We capture this property by $4t$ features where t denotes the maximum number of predicates that we consider simultaneously for creating the probability tables. In our experiments reported in the paper we took $t = 2$. These features are given below. The notation $\langle V_i \rangle_{i \in C}$ stands for average of quantity V_i given by $\frac{1}{|C|}\sum_i V_i$

for $t = 1$

$$f^{(1)}_{e\_prior} = \left\langle P^{(1)}_{e\_prior}(f_{il}) \right\rangle_{1 \le l \le p}$$

$$f^{(1)}_{a\_prior} = \left\langle P^{(1)}_{a\_prior}(pm(f_{il})) \right\rangle_{1 \le l \le p}$$

$$f^{(1)}_{ev\_prior} = \left\langle P^{(1)}_{ev\_prior}(f_{il} \mid \nu) \right\rangle_{1 \le l \le p}$$

$$f^{(1)}_{av\_prior} = \left\langle P^{(1)}_{av\_prior}(pm(f_{il}) \mid \nu) \right\rangle_{1 \le l \le p}$$

for $t = 2$

$$f^{(2)}_{e\_prior} = \left\langle P^{(2)}_{e\_prior}(f_{il_1} \wedge f_{il_2}) \right\rangle_{1 \le l_1 < l_2 \le p}$$

$$f^{(2)}_{a\_prior} = \left\langle P^{(2)}_{a\_prior}(pm(f_{il_1} \wedge f_{il_2})) \right\rangle_{1 \le l_1 < l_2 \le p}$$

$$f^{(2)}_{ev\_prior} = \left\langle P^{(2)}_{ev\_prior}(f_{il_1} \wedge f_{il_2} \mid \nu) \right\rangle_{1 \le l_1 < l_2 \le p}$$

$$f^{(2)}_{av\_prior} = \left\langle P^{(2)}_{av\_prior}(pm(f_{il_1} \wedge f_{il_2}) \mid \nu) \right\rangle_{1 \le l_1 < l_2 \le p}$$

The prior tables $P^{(t)}_r(|.)$ are created using the training data.

- *Logical Form and Environment*: As explained in the paper, we introduce the anchored mapping ξ to help in dealing with ellipsis. Therefore, we add a feature that maximizes the similarity between the anchored mapping $\xi(v_j)$ of a variable v_j and the new mapping $\xi_i(v_j)$. This is given by:

$$f_{ee} = \frac{1}{m}\sum_{j=1}^{m} \Delta(\xi(v_j), \xi_i(v_j)))$$

where Δ is a similarity score between objects $\xi(v_j)$ and $\xi_i(v_j)$. We compute this by taking $\Delta(o_1, o_2)$ = 0.5 $1\{o_1.\text{category} = o_2.\text{category}\}$

+ 0.5 fraction of common states value pairs.

- *Relationship Features*: Given all (ω_1, ω_2, r) pairs where $\omega_1, \omega_2 \in c_i$ and r is a spatial relationship between them. The relationship feature is given by:
$$f_{rel} = \langle y_{i,\omega_1,\omega_2} \rangle_{(\omega_1,\omega_2,r)}$$
where $y_{i,\omega_1,\omega_2} = 1$ if post-condition f_i contains a predicate $rel(o_1, o_2)$ where o_1, o_2 are the objects referred by description ω_1, ω_2 respectively.
- *Similarity Feature*: This is given by the Jaccard index of all the words in c_i and the words in the anchored lexical entry.
- *Transition Probabilities*: Given a logical form z_{i-1}, we can set priors on the logical form z_i. E.g., its unlikely that a logical form with post-condition $f_{i-1} = on(cup_1, counter_2)$ will be succeeded by logical form with post-condition $f_i = on(cup_1, counter_1)$. Further, the logical forms that can occur in the end state (c_i is the last frame node) are also restricted. We therefore, define 3 transition probability features to capture this:

$$f_{tr\_prior} = \langle P_{tr\_prior}(f_{i,l_1}, f_{i-1,l_2}) \rangle_{l_1,l_2}$$
$$f_{atr\_prior} = \langle P_{atr\_prior}(pm(f_{i,l_1}), pm(f_{i-1,l_2})) \rangle_{l_1,l_2}$$
$$f_{end} = \langle P_{end}(pm(f_{i,l})) \rangle_{l}$$

6 Assignment Problem

During inference, we want to generate logical forms $z = (\ell, \xi)$ for a given lexical entry $\ell = [\nu \Rightarrow (\lambda\vec{v}.S, \xi')]$. However the number of such logical forms are exponential in the number of variables in $\vec{v}$. Therefore, for practical reasons we only consider the optimum assignment given by $\arg\max_\xi \phi(z = (\ell, \xi), \cdots) \cdot \theta$. Note that we use slightly different notation from the paper, for reasons of brevity.

We convert this assignment problem into an optimization problem and then solve it approximately. To do so, we define 0-1 variables $y_{ij}; 1 \leq i \leq m; 1 \leq j \leq n$. Where m is the number of variables in $\vec{v}$ and n is the number of objects in the given environment e. Further $y_{ij} = 1$ iff variable v_i maps to the object o_j. Using this notation, the features described in Section 5 can be expressed as follows.

1. Language and Logical Form

$$f_{le} = \frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{n} (\max_\omega \rho(\omega, o_j)) y_{ij}$$
$$f_{recall} = \frac{1}{|\omega \in c|} \sum_{\omega \in c} \max_j (\sum_{i=1}^{m} \rho(\omega, o_j) y_{ij})$$

2. Logical Form

The environment prior terms can be easily expressed in a form which is polynomial in y_{ij}. For example, the feature $f^{(2)}_{e\text{-}prior}$ for the parametrized post-condition $(\texttt{state}\,\texttt{v}_1\,\texttt{water}) \wedge (\texttt{on}\,\texttt{v}_1\,\texttt{v}_2)$ can be expressed as:
$\sum_{r,s=1}^{m} \texttt{P}^{(2)}_{\texttt{e-prior}}((\texttt{state}\,\texttt{o}_\texttt{r}\,\texttt{water}) \wedge (\texttt{on}\,\texttt{o}_\texttt{r}\,\texttt{o}_\texttt{s})) y_{1r} y_{1s}$

3. Logical Form and Environment

Similarly, the f_{ee} term can be expressed as:
$f_{ee} = \frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{n} \Delta(\xi'(v_i), o_j)) y_{ij}$

4. Relationship Feature

For every $(\omega_1, \omega_2, r) \in c$ pair; we find the objects o_{j_1}, o_{j_2} referred to by these descriptions. Let the post-condition f contain atoms $f_1, f_2, \cdots f_l$ of the type $r(v_1, v_2)$ then for each such predicate, we consider the term $y_{1j_1} y_{2j_2}$.

5. Transition Probabilities

Transition probabilities are expressed similar to environment priors.

Dropping the higher order terms (generally small) and the $recall$ term(to simplify the optimization); we get a quadratic program of the form:

$$\max a^T x + x^T \mathbf{B} x$$
$$\mathbf{P}x \leq q$$

The linear constraints $\mathbf{P}x \leq q$ consists of $y_{ij} \in \{0, 1\}, \sum_j y_{ij} = 1$ and semantic constraints based on preconditions as given in the planner. E.g., for the post-condition $\texttt{on}(\texttt{v}_1, \texttt{v}_2)$, the planner preconditions tells that v_1 mus satisfy $IsGraspable(v_1)$; we therefore add these semantic constraints as inferred from the planner.

In this form, the assignment problem is nonconvex and does not necessarily admit a unique solution. While this can be solved by standard solvers such as AlgLib library; this optimization is quite slow and hence for practical reasons we drop the $\mathbf{B}$ term and solve the remaining linear program using a fast interior point solver after relaxation. The experiments in the paper are reported based on these approximations.