Language to action

CMSC 723 / LING 723 / INST 725

Hal Daumé III [he/him]

26 Nov 2019

Announcements, logistics

- Exam
  - Don’t forget to file regrade requests!
- Homework 4/5:
  - Everything out by now, deadline last day of classes (Dec 9)
  - Last part is EC
- Previous project grades – we’ll get them out by next class
- Plan for P5 deadline/exam period
  - One poster per team
    - Students should spend ~50% of time at their own poster, ~50% of time looking at others
    - Hal/TAs will go quickly through each poster, so be sure to have a 4min spiel prepared
  - Would you prefer in this room or in Iribe lobby (if I can get it)?

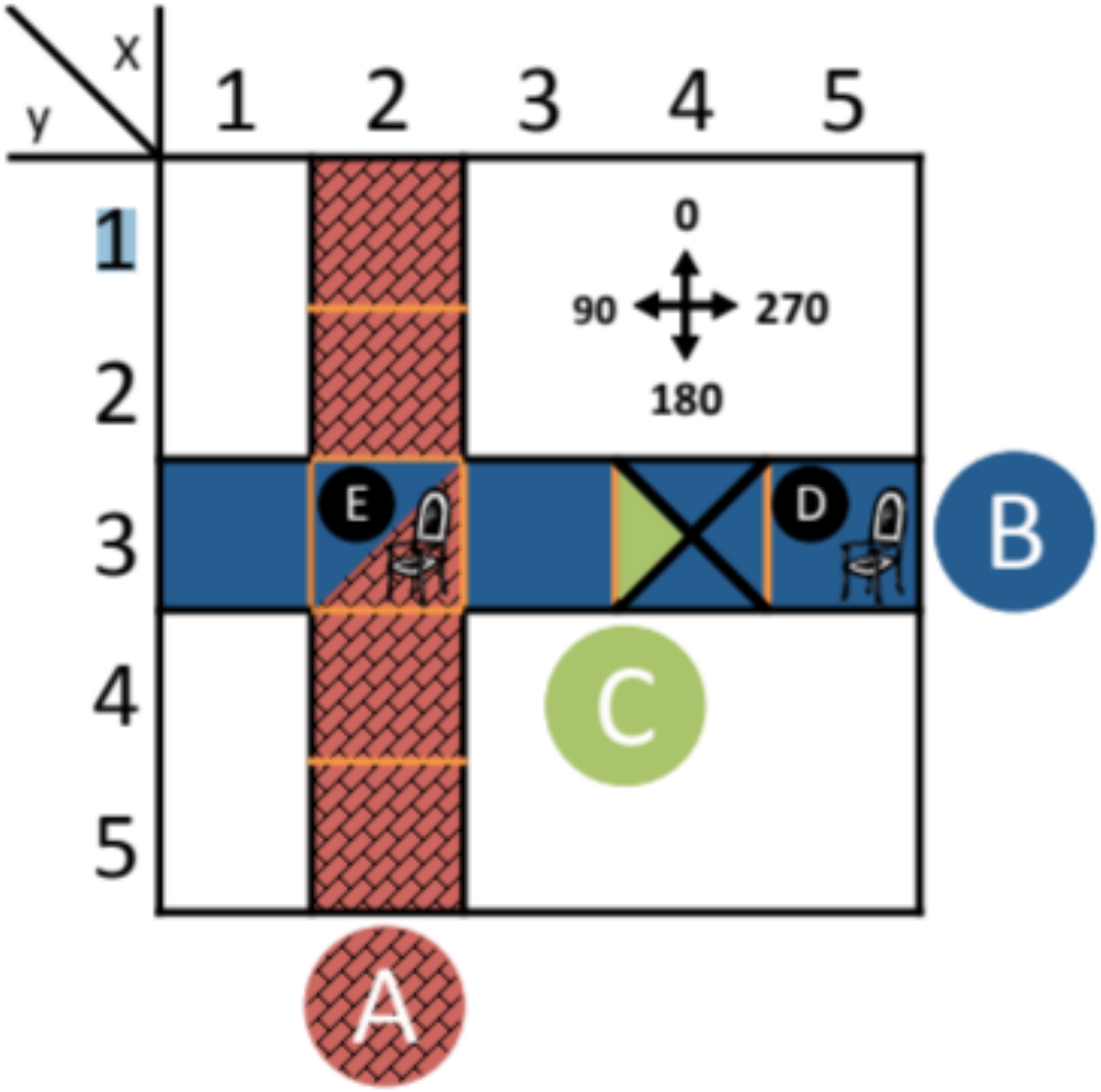
Last time

- Semantic parsing from denotations
  - Assume predicates in domain are known
  - Given (sentence, validation) pairs, learn a good parser
    - That maps sentence → logical form
    - With deterministic logical form → validation
- Grounding
  - How to learn correspondences between linguistic terms and non-linguistic items
  - Spatial language

Semantic parsing of instructior

Weakly Supervised Learning of Semantic Parsers for Mapping Instructions to Actions

Yoav Artzi and **Luke Zettlemoyer**



(a) chair
$\lambda x.chair(x)$

(b) hall
$\lambda x.hall(x)$

(c) the chair
$\iota x.chair(x)$

(d) you
you

(e) blue hall
$\lambda x.hall(x) \wedge blue(x)$

(f) chair in the intersection
$\lambda x.chair(x) \wedge$
$intersect(\iota y.junction(y), x)$

(g) in front of you
$\lambda x.in\_front\_of(you, x)$

Data/interaction

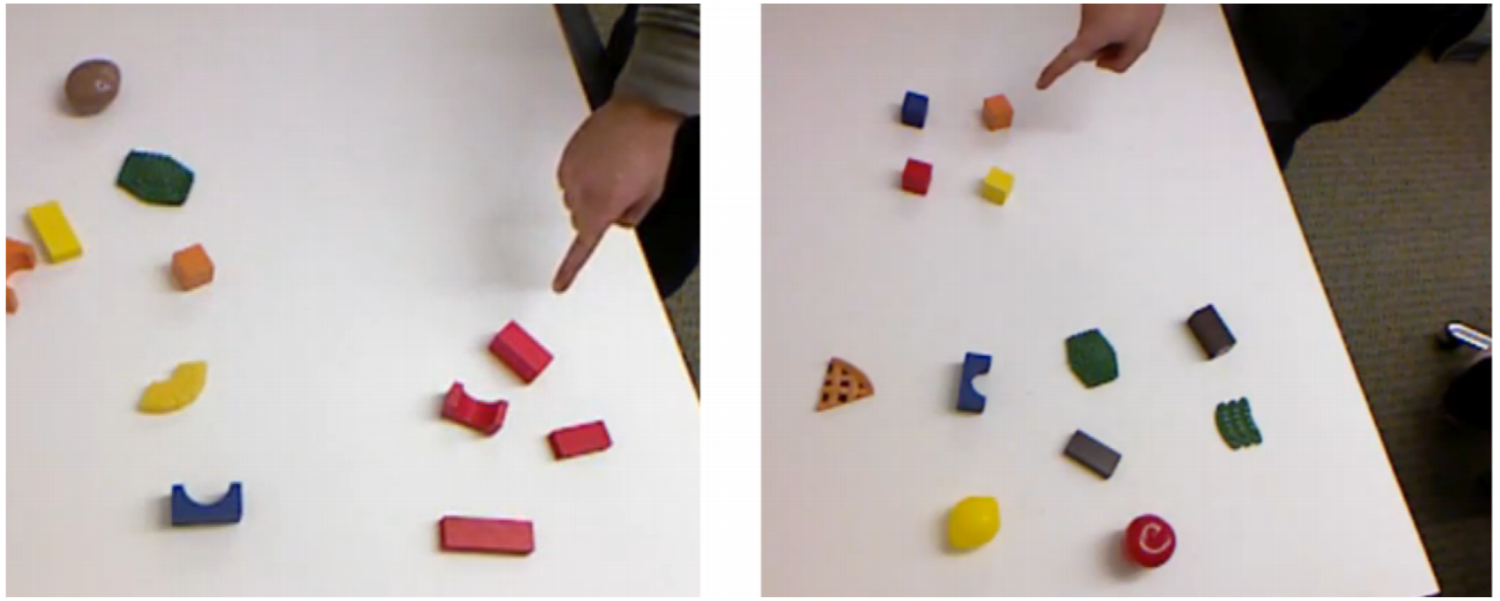


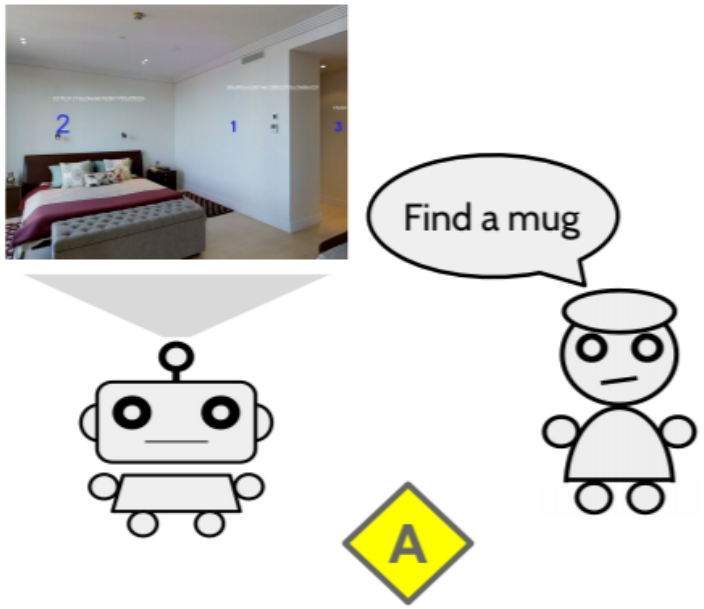
Figure 3. Example scenes presented on Mechanical Turk. Left: A scene that elicited the descriptions "here are some red things" and "these are various types of red colored objects", both labeled as $\lambda x.color(x, red)$. Right: A scene associated with sentence/meaning pairs such as "this toy is orange cube" and $\lambda x.color(x, orange) \wedge shape(x, cube)$.

Today

- Going directly from language to action

- Click start, point to search, and then click for files or folders.
- In the search results dialog box, on the tools menu, click folder options.
- In the folder options dialog box, on the view tab, under advanced settings, click *show hidden files and folders*, and then click to clear the *hide file extensions for known file types* check box.
- Click apply, and then click ok.
- In the search for files or folders named box, type msdownld.tmp.
- In the look in list, click my computer, and then click search now.
- In the search results pane, right-click msdownld.tmp and then click delete on the shortcut menu, a *confirm folder delete* message appears.
- Click yes.

Figure 1: A Windows troubleshooting article describing how to remove the "msdownld.tmp" temporary folder.



- Key idea: Formulate as reinforcement or imitation learning
- Key challenge: Getting data

To do RL or IL you must define….

- State space, S
- Action space, A
- Transition probabilities, P(s' I s, a)
- Initial state (distribution), s0

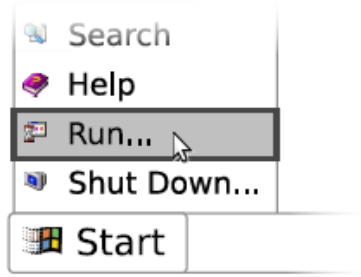
- And then one of:
  - Reward function R(s,a,s') // for RL, as proximal as possible
  - Expert policy pi*(s) → a // for IL

Setup

u: click **Run**, and press **OK** after typing **secpol.msc** in the **open** box.

$\vec{a}$: c: *left-click* R: [**Run**...]

$\mathcal{E}$:

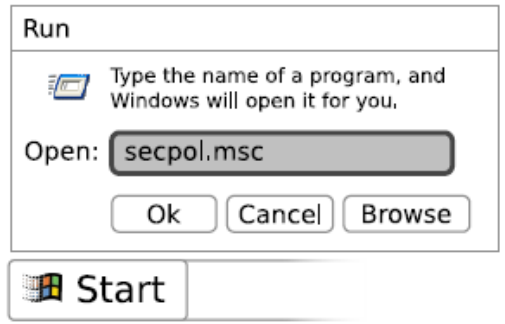


---

u: click **Run**, and press **OK** after typing **secpol.msc** in the **open** box.

$\vec{a}$: *left-click* **Run**... c: *type-into* R: [**open** *"secpol.msc"*]

$\mathcal{E}$:

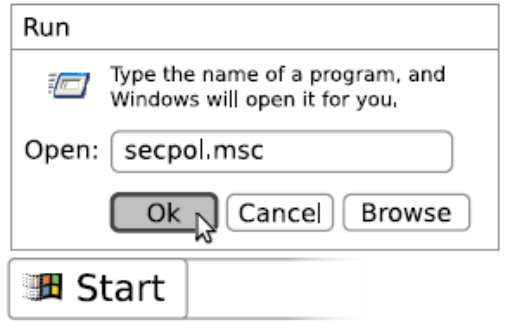


---

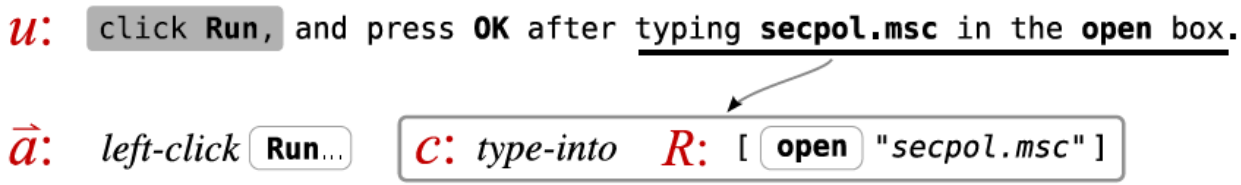
u: click **Run**, and press **OK** after typing **secpol.msc** in the **open** box.

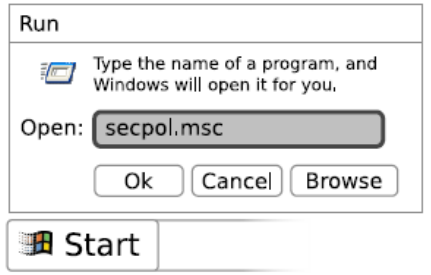
$\vec{a}$: *left-click* **Run**... *type-into* **open** *"secpol.msc"* c: *left-click* R: [**OK**]

$\mathcal{E}$:



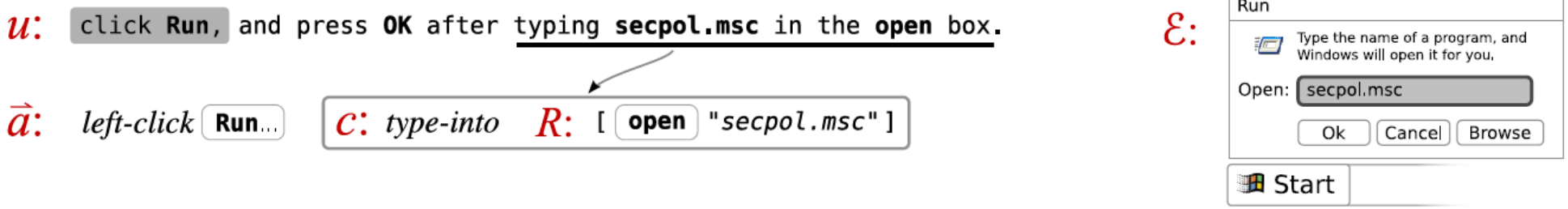
Actions





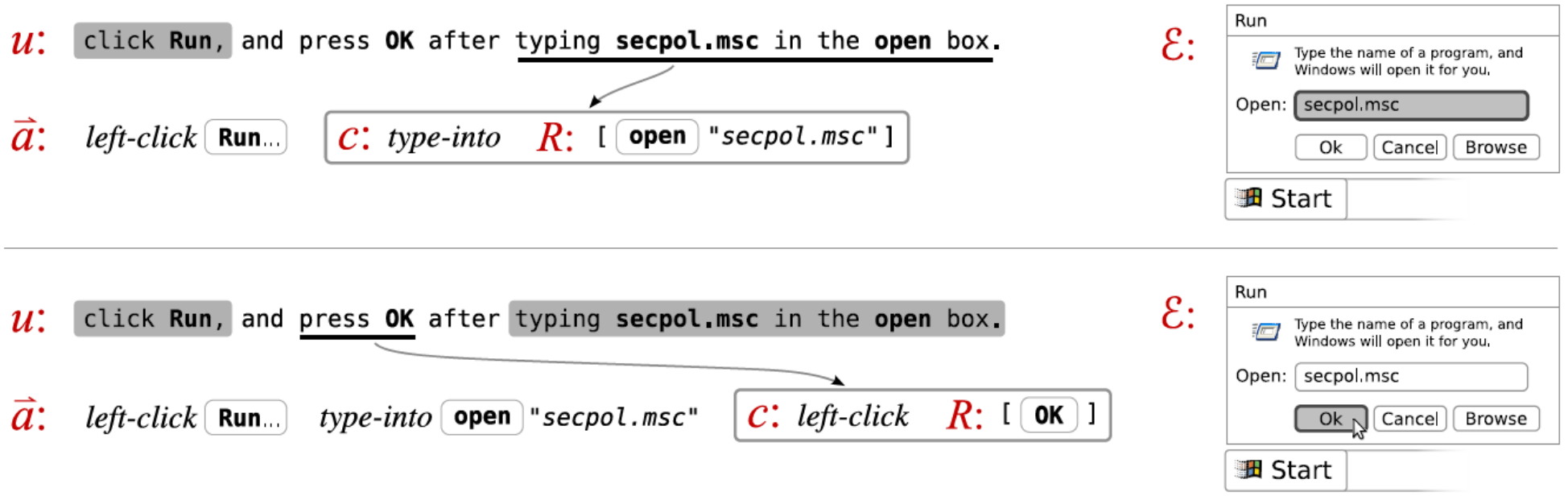
An action $a = (c, R, W')$ encompasses a *command* c, the command's *parameters* R, and the words W' specifying c and R. Elements of R refer to *objects* available in the *environment state*, as described below. Some parameters can also refer to words in document d. Additionally, to account for words that do not describe any actions, c can be a null command.

States



State To predict actions sequentially, we need to track the state of the document-to-actions mapping over time. A *mapping state* s is a tuple $(\mathcal{E}, d, j, W)$, where $\mathcal{E}$ refers to the current environment state; j is the index of the sentence currently being interpreted in document d; and W contains words that were mapped by previous actions for the same sentence. The mapping state s is observed after each action.

Transitions



state for d. Performing action a in state $s = (\mathcal{E}, d, j, W)$ leads to a new state s' according to distribution $p(s'|s, a)$, defined as follows: $\mathcal{E}$ transitions according to $p(\mathcal{E}'|\mathcal{E}, c, R)$, W is updated with a's selected words, and j is incremented if all words of the sentence have been mapped. For the applications we consider in this work, environment state transitions, and consequently mapping state transitions, are deterministic.

Reward

Training During training, we are provided with a set D of documents, the ability to sample from the transition distribution, and a *reward function* $r(h)$. Here, $h = (s_0, a_0, \ldots, s_{n-1}, a_{n-1}, s_n)$ is a *history* of states and actions visited while interpreting one document. $r(h)$ outputs a real-valued score that correlates with correct action selection.[3] We consider both immediate reward, which is available after each action, and delayed reward, which does not provide feedback until the last action. For example, *task completion* is a delayed reward that produces a positive value after the final action only if the task was completed successfully. We will also demonstrate how manually annotated action sequences can be incorporated into the reward.

Model

- Linear-softmax policy: $$p(a|s;\theta) = \frac{e^{\theta \cdot \phi(s,a)}}{\sum_{a'} e^{\theta \cdot \phi(s,a')}}$$

- Features:

Notation	
o	Parameter referring to an environment object
L	Set of object class names (e.g. "button")
V	Vocabulary
Features on W and object o	
Test if o is visible in s	
Test if o has input focus	
Test if o is in the foreground	
Test if o was previously interacted with	
Test if o came into existence since last action	
Min. edit distance between $w \in W$ and object labels in s	
Features on words in W, command c, and object o	
$\forall c' \in C, w \in V$: test if $c' = c$ and $w \in W$	
$\forall c' \in C, l \in L$: test if $c' = c$ and l is the class of o	

Policy gradient:

for $i = 1 \ldots T$ **do**
  foreach $d \in D$ **do**
    Sample history $h \sim p(h|\theta)$ where
      $h = (s_0, a_0, \ldots, a_{n-1}, s_n)$ as follows:
    for $t = 0 \ldots n-1$ **do**
      Sample action $a_t \sim p(a|s_t;\theta)$
      Execute a_t on state s_t: $s_{t+1} \sim p(s|s_t, a_t)$
    end
    $\Delta \leftarrow \sum_t \left(\phi(s_t, a_t) - \sum_{a'} \phi(s_t, a') p(a'|s_t;\theta)\right)$
    $\theta \leftarrow \theta + r(h)\Delta$
  end
end

Data

	Windows
Total # of documents	128
Total # of words	5562
Vocabulary size	610
Avg. words per sentence	9.93
Avg. sentences per document	4.38
Avg. actions per document	10.37

On the ***tools*** menu, click ***internet options***
Click ***tools***, and then click ***internet options***
Click ***tools***, and then choose ***internet options***
Click ***internet options*** on the ***tools*** menu
In internet explorer, click ***internet options*** on the ***tools*** menu
On the ***tools*** menu in internet explorer, click ***internet options***

Figure 4: Variations of "click internet options on the tools menu" present in the Windows corpus.

Reward functions and "reward" functions

Environment reward

Instead, we rely on a noisy method of checking whether execution can proceed from one sentence to the next: at least one word in each sentence has to correspond to an object in the environment.[6] For instance, in the sentence from Figure 2 the word "Run" matches the *Run...* menu item. If no words in a sentence match a current environment object, then one of the previous sentences was analyzed incorrectly. In this case, we assign the history a reward of -1. This reward is not guaranteed to penalize all incorrect histories, because there may be false positive matches between the sentence and the environment. When at least one word matches, we assign a positive reward that linearly increases with the percentage of words assigned to non-null commands, and linearly decreases with the number of output actions. This reward signal encourages analyses that interpret all of the words without producing spurious actions.

Imitation reward

tated data and environment feedback. Consider the case when every training document $d \in D$ is annotated with its correct sequence of actions, and state transitions are deterministic. Given these examples, it is straightforward to construct a reward function that connects policy gradient to maximum likelihood. Specifically, define a reward function $r(h)$ that returns one when h matches the annotation for the document being analyzed, and zero otherwise. Policy gradient performs stochastic gradient ascent on the objective from equation 2, performing one update per document. For document d, this objective becomes:

$$E_{p(h|\theta)}[r(h)] = \sum_h r(h)p(h|\theta) = p(h_d|\theta),$$

where h_d is the history corresponding to the annotated action sequence. Thus, with this reward policy gradient is equivalent to stochastic gradient ascent with a maximum likelihood objective.

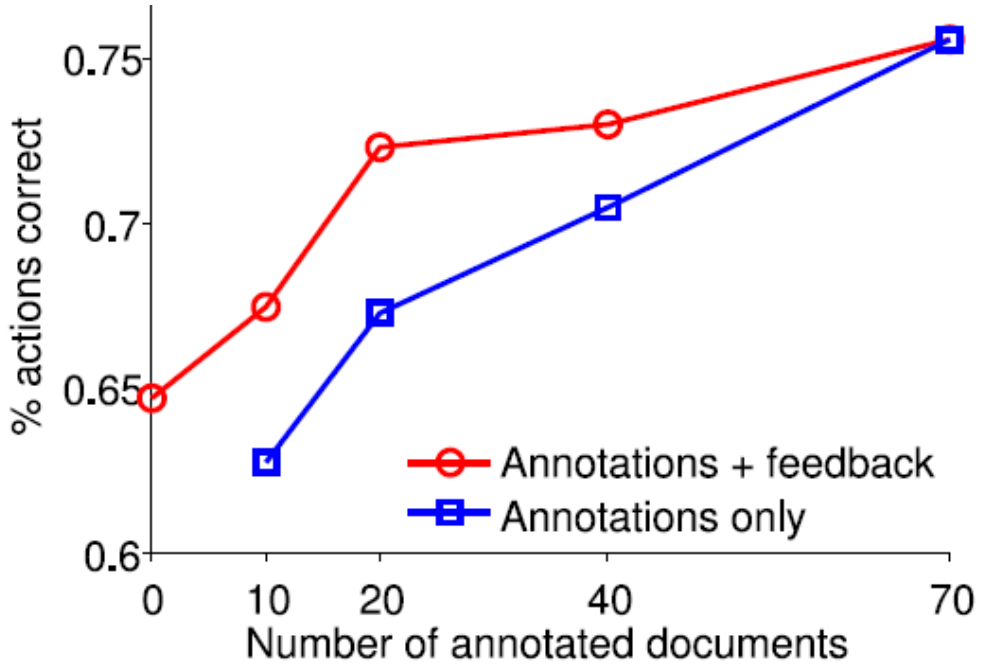
Evaluation

- Baselines:
  - Full supervision (imitation alone)
  - Partial supervision (imitation + reinforcement)
  - Random/majority – match text

Evaluation Metrics For evaluation, we compare the results to manually constructed sequences of actions. We measure the number of correct actions, sentences, and documents. An action is correct if it matches the annotations in terms of command and parameters. A sentence is correct if all of its actions are correctly identified, and analogously for documents.[11] Statistical significance is measured with the sign test.

Results

	Windows			
	Action	Sent.	Doc.	Word
Random baseline	0.128	0.101	0.000	—
Majority baseline	0.287	0.197	0.100	—
Environment reward	∗ 0.647	∗ 0.590	∗ 0.375	0.819
Partial supervision	◇ 0.723	∗ 0.702	0.475	0.989
Full supervision	◇ 0.756	0.714	0.525	0.991



Robust Generalization via
Leveraging External Knowledge Sources

AI agents with the ability to find and leverage external knowledge sources can self-learn to accomplish tasks that they cannot accomplish on their own.

How do humans leverage assistance?

This is Bob. He is on a vacation in a foreign country.

Bob wants to ride a scooter to the Iribe Center--a famous tourist attraction.

Unfortunately, he gets lost. The map does not seem to help much. His cellphone also does not work.

How do humans leverage assistance?

Lucky for Bob, he met Taha, a local.

Bob explains his situation and the purpose of his trip to Taha.

Taha calms Bob down and shows him the way to the Iribe Center.

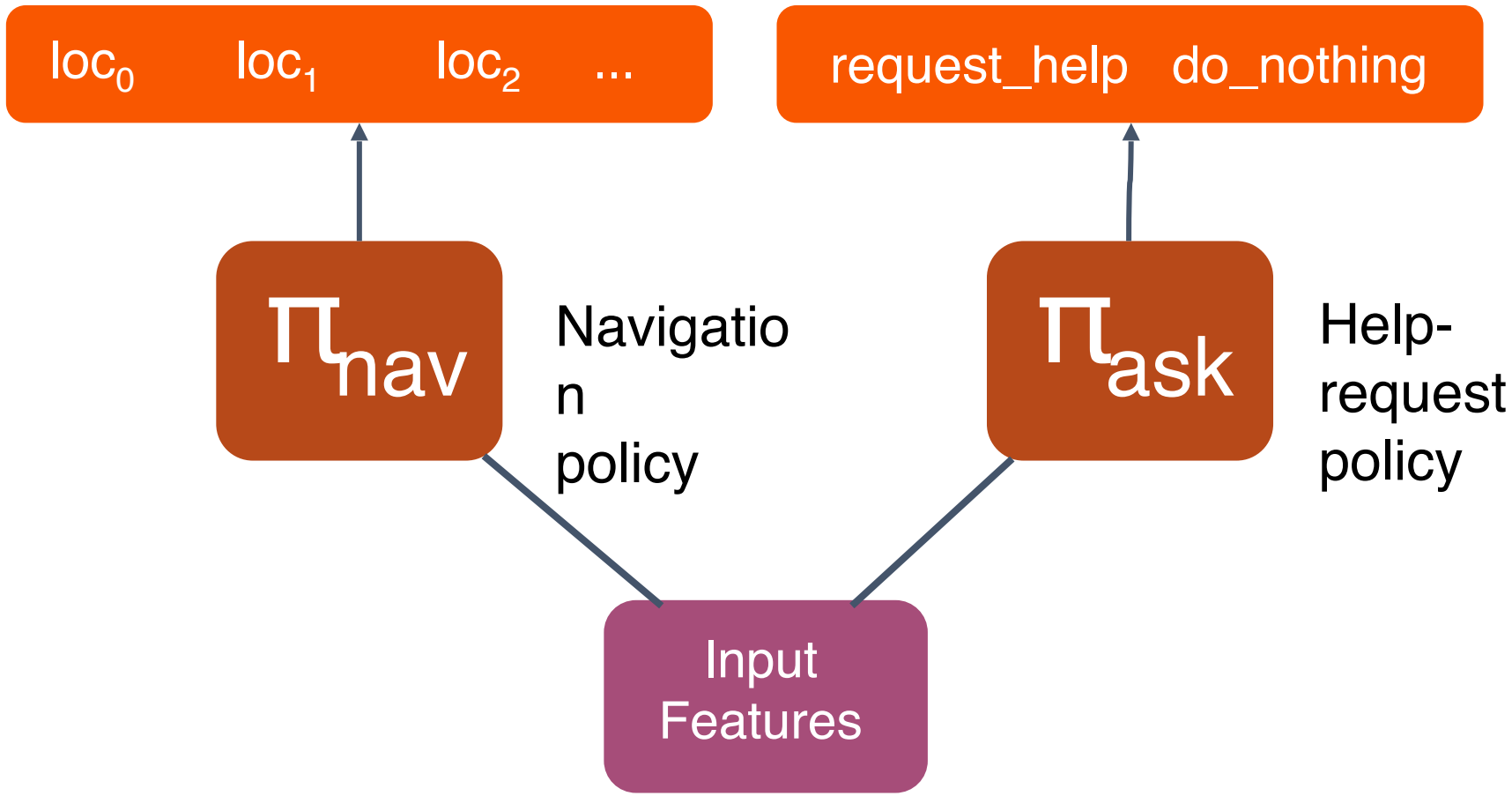
Help! Anna!

Photorealistic Vision-based Navigation with Natural Multimodal Assistance

[hanna_env.mp4]

[hanna_task.mp4]

Agent Policies



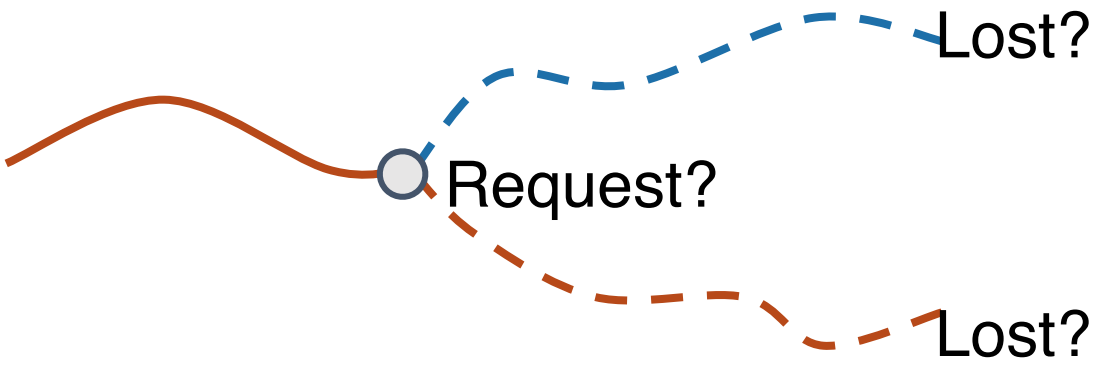
When to request help?

- Uncertainty-based (Nguyen et al., 2019):

  Entropy[Predict_dist] - Entropy[Uniform_dist] < ε

Retrospective Imitation Learning

How to compute the teacher actions efficiently?

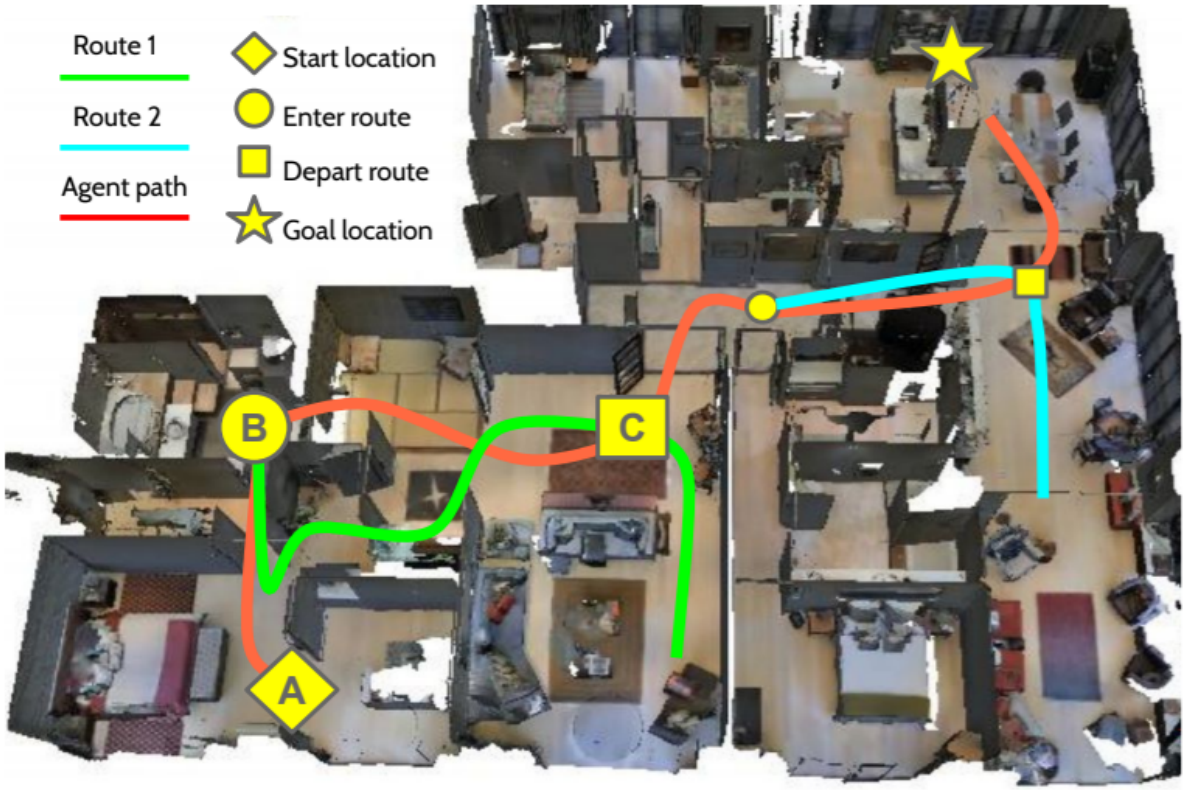


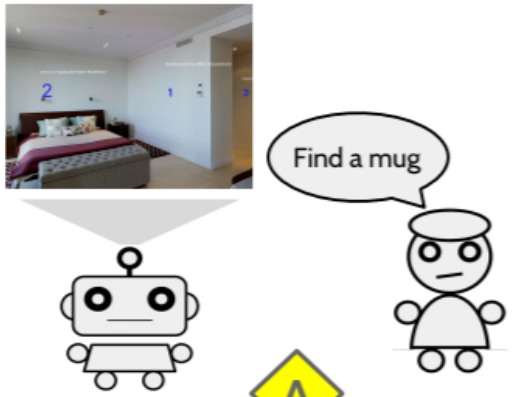
Online teacher: computes ground-truth for each time step on-the-fly
⇒ Expensive, requires O(T) roll-outs

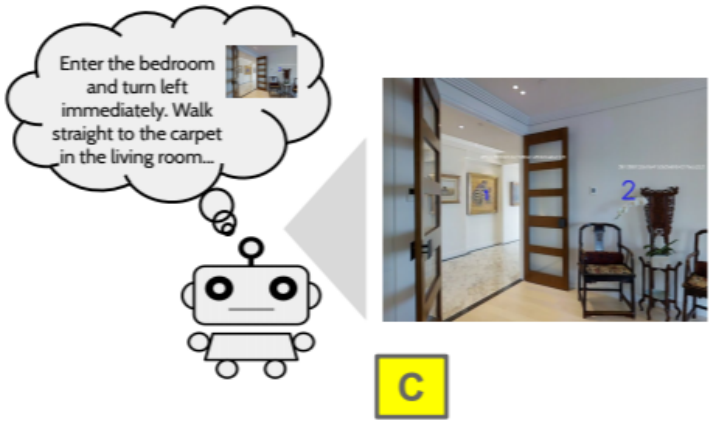
optimal for binary action space

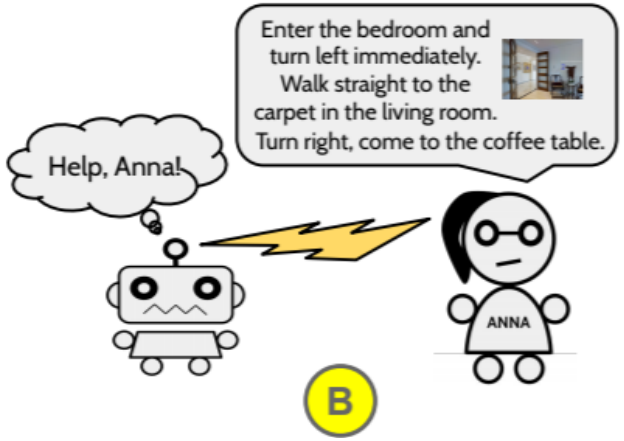
[hanna_shortwalk.mp4]

Example setup

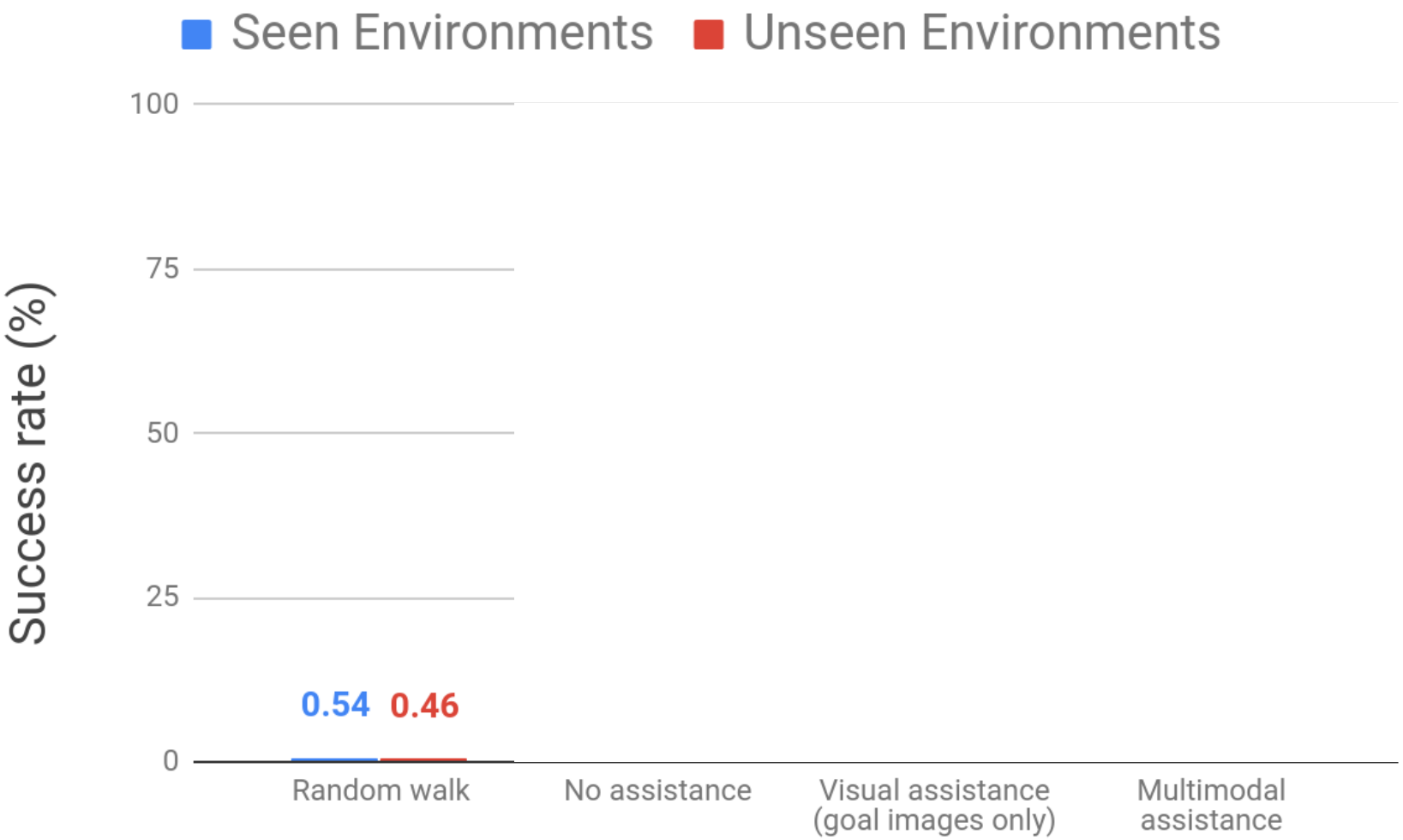




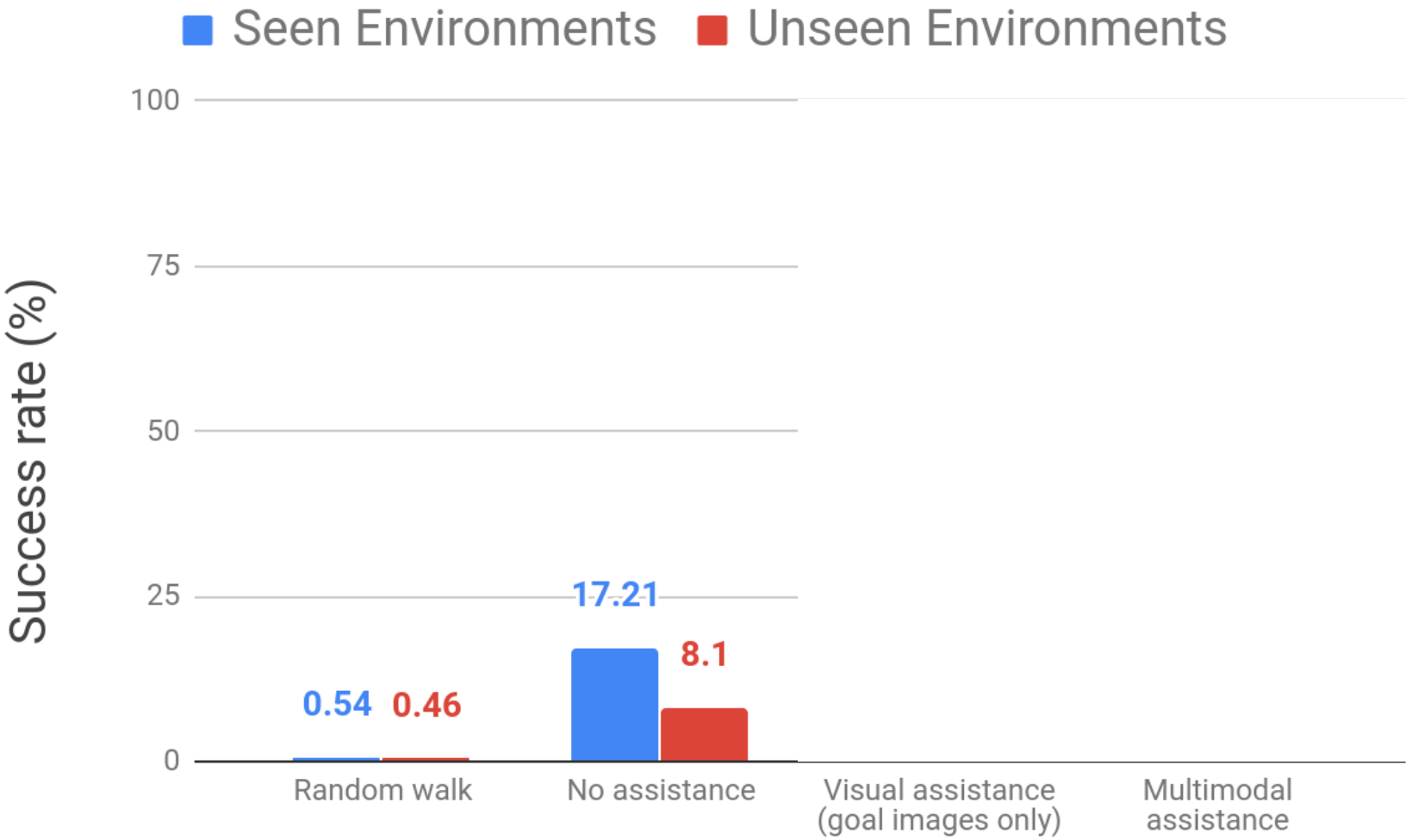




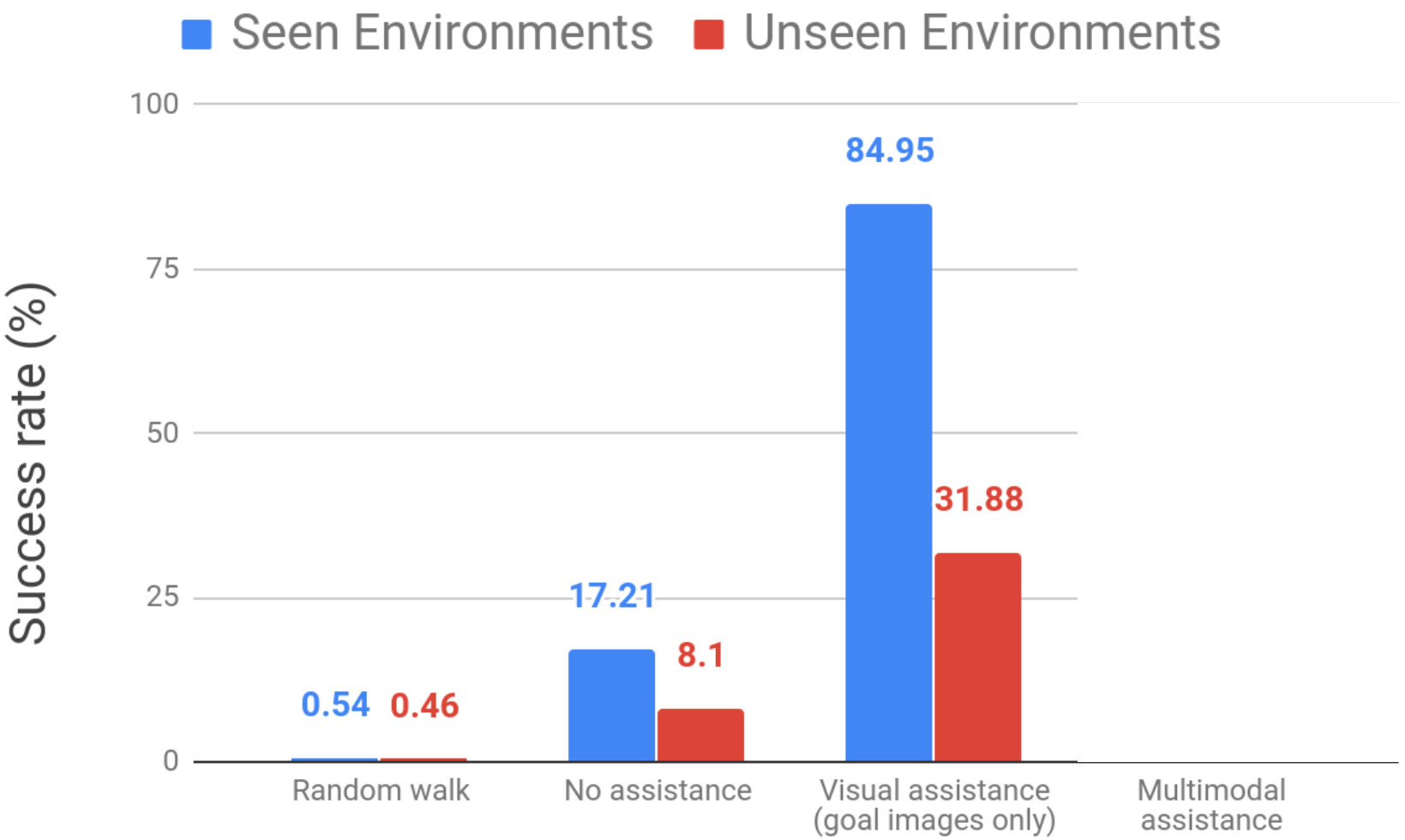
Main Results



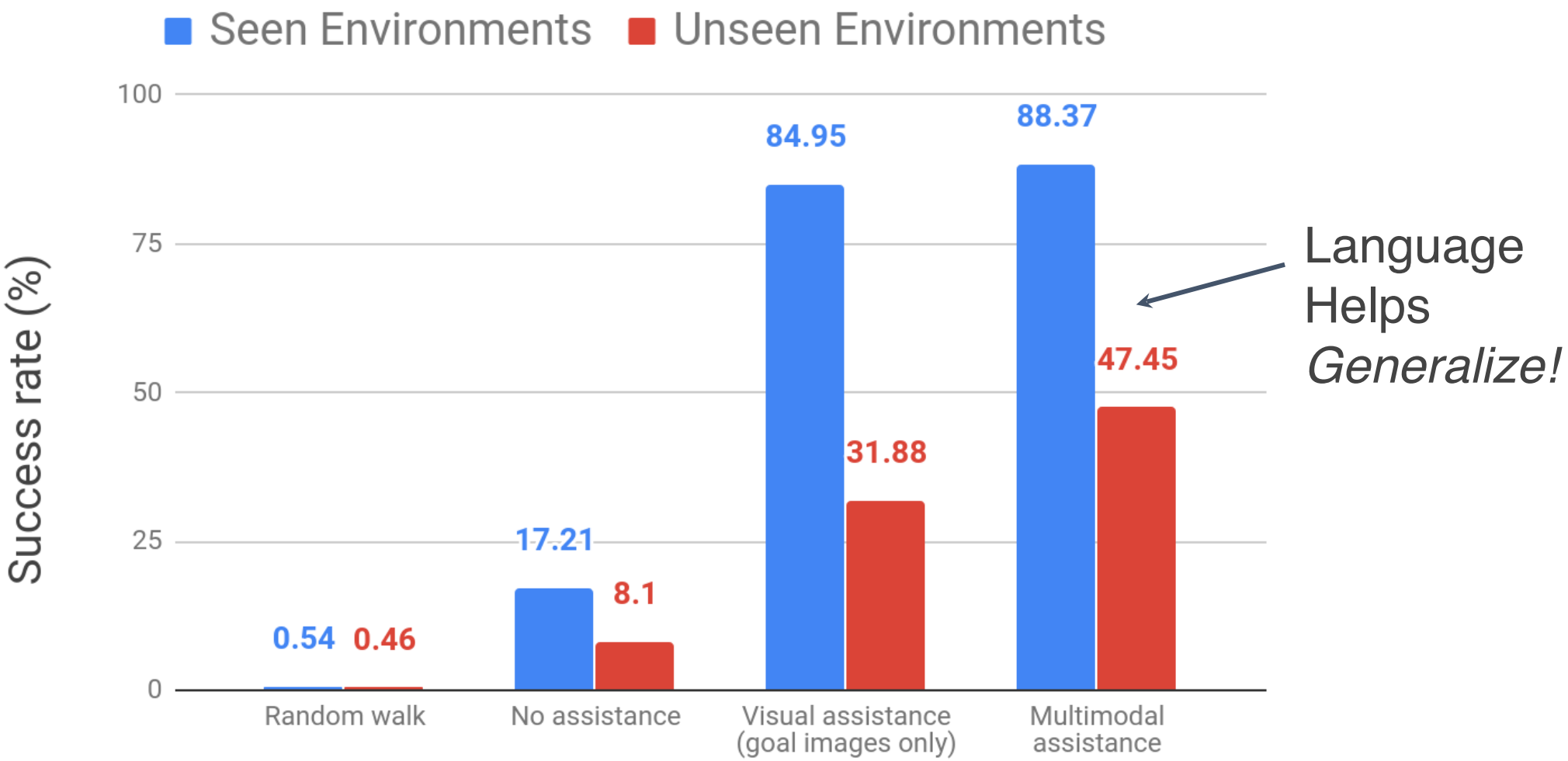
Main Results



Main Results



Main Results



Comparison of Request Policies

	SeenEnv	
$\hat{\pi}_{\text{ask}}$	SR ↑ (%)	Requests/ task ↓
NoAsk	17.21	0.0
RandomAsk	82.71	4.3
AskEvery5	87.39	3.4
Learned (ours)	**88.37**	**2.9**

Comparison of Request Policies

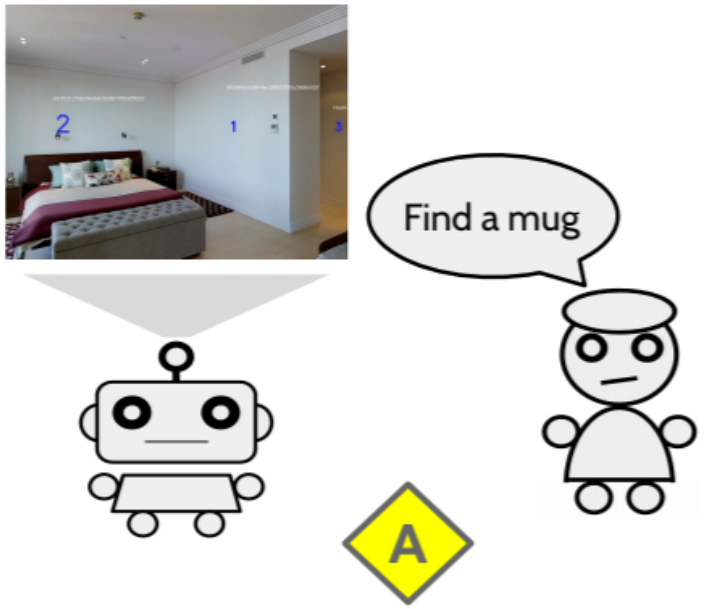
$\hat{\pi}_{\text{ask}}$	SeenEnv SR ↑ (%)	SeenEnv Requests/ task ↓	UnseenAll SR ↑ (%)	UnseenAll Requests/ task ↓
NoAsk	17.21	0.0	8.10	0.0
RandomAsk	82.71	4.3	37.05	6.8
AskEvery5	87.39	3.4	34.42	7.1
Learned (ours)	**88.37**	**2.9**	**47.45**	**5.8**

Today

- Going directly from language to action

- Click start, point to search, and then click for files or folders.
- In the search results dialog box, on the tools menu, click folder options.
- In the folder options dialog box, on the view tab, under advanced settings, click *show hidden files and folders*, and then click to clear the *hide file extensions for known file types* check box.
- Click apply, and then click ok.
- In the search for files or folders named box, type msdownld.tmp.
- In the look in list, click my computer, and then click search now.
- In the search results pane, right-click msdownld.tmp and then click delete on the shortcut menu, a *confirm folder delete* message appears.
- Click yes.

Figure 1: A Windows troubleshooting article describing how to remove the "msdownld.tmp" temporary folder.



- Key idea: Formulate as reinforcement or imitation learning
- Key challenge: Getting data