Notation	Definition
$\Delta(\mathcal{U})$	Space of all distributions over a set $\mathcal{U}$
$\texttt{unf}(\mathcal{U})$	Denotes the uniform distribution over a set $\mathcal{U}$
$\|.\|_p$	p-norm
$\mathrm{D}_{\mathrm{KL}}(Q_1(x \mid y) \mid\mid Q_2(x \mid y))$	KL-divergence between two distributions $Q_1(\cdot \mid y)$ and $Q_2(\cdot \mid y)$ over a countable set $\mathcal{X}$. Formally, $\mathrm{D}_{\mathrm{KL}}(Q_1(x \mid y) \mid\mid Q_2(x \mid y)) = \sum_{x \in \mathcal{X}} Q_1(x \mid y) \ln \frac{Q_1(x\mid y)}{Q_2(x\mid y)}$.
$\texttt{supp}\ Q(x)$	Support of a distribution $Q \in \Delta(\mathcal{X})$. Formally, $\texttt{supp}Q(x) = \{x \in \mathcal{X} \mid Q(x) > 0\}$.
$\mathbb{N}$	Set of natural numbers
$\mathcal{S}$	State space
s	A single state in $\mathcal{S}$
$\mathcal{A}$	Finite action space
a	a single action in $\mathcal{A}$
$\mathcal{D}$	Set of all possible descriptions and requests
d	A single description or request
$T : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$	Transition function with $T(s' \mid s, a)$ denoting the probability of transitioning to state s' given state s and action a.
$\mathcal{R}$	Family of reward functions
$R : \mathcal{S} \times \mathcal{A} \to [0, 1]$	Reward function with $R(s, a)$ denoting the reward for taking action a in state s
H	Horizon of the problem denoting the number of actions in a single episode.
e	An execution $e = (s_1, a_1, s_2, \cdots, s_H, a_H)$ describing states and actions in an episode.
$q = (R, d, s_1)$	A single task comprising of reward function R, request d and start state s_1
$\mathbb{P}^\star(q)$	Task distribution defined by the world
$\mathbb{P}^\star(e, R, s, d)$	Joint distribution over executions and task (see Equation 2).
$\mathbb{P}_T(d \mid e)$	Teacher model denoting distribution over descriptions d for a given execution e.
Θ	Set of all parameters of agent's policy.
θ	Parameters of agent's policy. Belongs to the set Θ.
$\pi_\theta(a \mid s, d)$	Agent's policy denoting the probability of action a given state s, description d, and parameters θ.

Table 4. List of common notations and their definitions.

Appendix: Interactive Learning from Activity Description

The appendix is organized as follows:

- Statement and proof of theoretical guarantees for ADEL (Appendix A);
- Settings of the two problems we conduct experiments on (Appendix B);
- A practical implementation of the ADEL algorithm that we use for experimentation (Appendix C);
- Training details including model architecture and hyperparameters (Appendix D);
- Qualitative examples (Appendix E).

We provide a list of notations in Table 4 on page 14.

A. Theoretical Analysis of ADEL

In this section, we provide a theoretical justification for an epoch-version of ADEL for the case of $H = 1$. We prove consistency results showing ADEL learns a near-optimal policy, and we also derive the convergence rate under the assumption that we perform maximum likelihood estimation optimally and the teacher is consistent. We call a teacher model $\mathbb{P}_T(d \mid e)$

to be consistent if for every execution e and description d we have $\mathbb{P}_T(d \mid e) = \mathbb{P}^\star(d \mid e)$. Recall that the conditional distribution $\mathbb{P}^\star(d \mid e)$ is derived from the joint distribution defined in Equation 2. We will use superscript * to denote all probability distributions that are derived from this joint distribution.

We start by writing the epoch-version of ADEL in Algorithm 4 for an arbitrary value of H. The epoch version of ADEL runs an outer loop of epochs (line 3-10). The agent model is updated only at the end of an epoch. In the inner loop (line 5-9), the agent samples a batch using the teacher model and the agent model. This is used to update the model at the end of the epoch.

At the start of the n^{th} epoch, our sampling scheme in line 6-9 defines a procedure to sample $(\hat{e}, \hat{d})$ from a distribution D_n that remains fixed over this whole epoch. To define D_n, we first define $\mathbb{P}_n(e) = \mathbb{E}_{(R,d,s_1)\sim\mathbb{P}^\star(q)}\left[\mathbb{P}_n(e \mid s_1, d)\right]$ where we use the shorthand $\mathbb{P}_n(e \mid s_1, d)$ to refer to $\mathbb{P}_{\pi_{\theta_n}}(e \mid s_1, d)$. Note that $\hat{e} \sim \mathbb{P}_n(e)$ in line 7. As $\hat{d} \sim \mathbb{P}^\star(d \mid \hat{e})$, therefore, we arrive at the following form of D_n:

$$D_n(\hat{e}, \hat{d}) = \mathbb{P}^\star(\hat{d} \mid \hat{e})\mathbb{P}_n(\hat{e}). \tag{8}$$

We will derive our theoretical guarantees for $H = 1$. This setting is known as the contextual bandit setting (Langford & Zhang, 2008a), and while simpler than general reinforcement learning setting, it captures a large non-trivial class of problems. In this case, an execution $e = [s_1, a_1]$ can be described by the start state s_1 and a single action $a_1 \in \mathcal{A}$ taken by the agent. Since there is a single state and action in any execution, therefore, for cleaner notations we will drop the subscript and simply write s, a instead of s_1, a_1. For convenience, we also define a few extra notations. Firstly, we define the marginal distribution $D_n(s, \hat{d}) = \sum_{a' \in \mathcal{A}} D_n([s, a'], d)$. Secondly, let $\mathbb{P}^\star(s)$ be the marginal distribution over start state s given by $\mathbb{E}_{(R,d,s_1)\sim\mathbb{P}^\star(q)}[\mathbf{1}\{s_1 = s\}]$. We state some useful relations between these probability distributions in the next lemma.

Algorithm 4 EPOCHADEL: Epoch Version of ADEL. We assume the teacher is consistent, i.e., $\mathbb{P}_T(d \mid e) = \mathbb{P}^\star(d \mid e)$ for every (d, e).

1: **Input**: teacher model $\mathbb{P}^\star(d \mid e)$ and task distribution model $\mathbb{P}^\star(q)$.
2: Initialize agent policy $\pi_{\theta_1} : \mathcal{S} \times \mathcal{D} \rightarrow \texttt{unf}(\mathcal{A})$
3: **for** $n = 1, 2, \cdots, N$ **do**
4: $\quad \mathcal{B} = \emptyset$
5: $\quad$ **for** $m = 1, 2, \cdots, M$ **do**
6: $\quad\quad$ World samples $q = (R, d^\star, s_1) \sim \mathbb{P}^\star(\cdot)$
7: $\quad\quad$ Agent generates $\hat{e} \sim \mathbb{P}_{\pi_{\theta_n}}(\cdot \mid s_1, d^\star)$
8: $\quad\quad$ Teacher generates description $\hat{d} \sim \mathbb{P}^\star(\cdot \mid \hat{e})$
9: $\quad\quad$ $\mathcal{B} \leftarrow \mathcal{B} \cup \left\{\left(\hat{e}, \hat{d}\right)\right\}$
10: $\quad$ Update agent policy using batch updates:

$$\theta_{n+1} \leftarrow \arg\max_{\theta' \in \Theta} \sum_{(\hat{e},\hat{d})\in\mathcal{B}} \sum_{(s,a_s)\in\hat{e}} \log \pi_{\theta'}(a_s \mid s, \hat{d})$$

where a_s is the action taken by the agent in state s in execution $\hat{e}$.
return π_θ

Lemma 2. *For any $n \in \mathbb{N}$, we have:*

$$\mathbb{P}_n(e := [s, a]) = \mathbb{P}^\star(s)\mathbb{P}_n(a \mid s), \text{ where } \mathbb{P}_n(a \mid s) := \sum_d \mathbb{P}^\star(d \mid s)\mathbb{P}_n(a \mid s, d). \tag{9}$$

Proof. We first compute the marginal distribution $\sum_{a' \in \mathcal{A}} \mathbb{P}_n(e' := [s, a'])$ over s:

$$\sum_{a' \in \mathcal{A}} \mathbb{P}_n(e' := [s, a']) = \sum_{a' \in \mathcal{A}} \sum_{R,d} \mathbb{P}^\star(R, d, s)\mathbb{P}_n(a' \mid s, d) = \sum_{R,d} \mathbb{P}^\star(R, d, s) = \mathbb{P}^\star(s).$$

Next we compute the conditional distribution $\mathbb{P}_n(a \mid s)$ as shown:

$$\mathbb{P}_n(a \mid s) = \frac{\mathbb{P}_n([s, a])}{\sum_{a' \in \mathcal{A}} \mathbb{P}_n([s, a'])} = \sum_{R,d} \frac{\mathbb{P}^\star(R, d, s)\mathbb{P}_n(a \mid s, d)}{\mathbb{P}^\star(s)} = \sum_d \frac{\mathbb{P}^\star(s, d)\mathbb{P}_n(a \mid s, d)}{\mathbb{P}^\star(s)} = \sum_d \mathbb{P}^\star(d \mid s)\mathbb{P}_n(a \mid s, d).$$

This also proves $\mathbb{P}_n([s, a]) = \mathbb{P}^\star(s)\mathbb{P}_n(a \mid s)$. $\square$

For $H = 1$, the update equation in line 10 solves the following optimization equation:

$$\max_{\theta' \in \Theta} J_n(\theta) \qquad \text{where} \quad J_n(\theta) := \sum_{(\hat{e} := [s,a], \hat{d}) \in \mathcal{B}} \ln \pi_{\theta'}(a \mid s, \hat{d}). \tag{10}$$

Here $J_n(\theta)$ is the empirical objective whose expectation over draws of batches is given by:

$$\mathbb{E}[J_n(\theta)] = \mathbb{E}_{(\hat{e}=[s,a],d) \sim D_n} \left[\ln \pi_\theta(a \mid s, d)\right].$$

As this is negative of the cross entropy loss, the Bayes optimal value would be achieved for $\pi_\theta(a \mid s, d) = D_n(a \mid s, d)$ for all $a \in \mathcal{A}$ and every $(s, d) \in \mathtt{supp} D_n(s, d)$. We next state the form of this Bayes optimal model and then state our key realizability assumption.

Lemma 3. *Fix $n \in \mathbb{N}$. For every $(s, d) \in \mathtt{supp} D_n(s, d)$ the value of the Bayes optimal model $D_n(a \mid s, d)$ at the end of the n^{th} epoch is given by:*

$$D_n(a \mid s, d) = \frac{\mathbb{P}^\star(d \mid [s, a])\mathbb{P}_n(a \mid s)}{\sum_{a' \in \mathcal{A}} \mathbb{P}^\star(d \mid [s, a'])\mathbb{P}_n(a' \mid s)}.$$

Proof. The Bayes optimal model is given by $D_n(a \mid s, d)$ for every $(s, d) \in \mathtt{supp} D_n(s, d)$. We compute this using Bayes' theorem.

$$D_n(a \mid s, d) = \frac{D_n([s, a], d)}{\sum_{a' \in \mathcal{A}} D_n([s, a'], d)} = \frac{\mathbb{P}^\star(d \mid [s, a])\mathbb{P}_n([s, a])}{\sum_{a' \in \mathcal{A}} \mathbb{P}^\star(d \mid [s, a'])\mathbb{P}_n([s, a'])} = \frac{\mathbb{P}^\star(d \mid [s, a])\mathbb{P}_n(a \mid s)}{\sum_{a' \in \mathcal{A}} \mathbb{P}^\star(d \mid [s, a'])\mathbb{P}_n(a' \mid s)}.$$

The last equality above uses Lemma 2. □

In order to learn the Bayes optimal model, we need our policy class to be expressive enough to contain this model. We formally state this *realizability* assumption below.

Assumption 1 (Realizability)**.** *For every $\theta \in \Theta$, there exists $\theta' \in \Theta$ such that for every start state s, description d we have:*

$$\forall a \in \mathcal{A}, \quad \pi_{\theta'}(a \mid s, d) = \frac{\mathbb{P}^\star(d \mid [s, a])Q_\theta(a \mid s)}{\sum_{a' \in \mathcal{A}} \mathbb{P}^\star(d \mid [s, a'])Q_\theta(a' \mid s)}, \quad \textit{where} \quad Q_\theta(a \mid s) = \sum_{d'} \mathbb{P}^\star(d' \mid s)\pi_\theta(a \mid s, d').$$

We can use the realizability assumption along with convergence guarantees for log-loss to state the following result:

Theorem 4 (Theorem 21 of (Agarwal et al., 2020))**.** *Fix $m \in \mathbb{N}$ and $\delta \in (0, 1)$. Let $\{(d^{(i)}, e^{(i)} = [s^{(i)}, a^{(i)}]\}_{i=1}^m$ be i.i.d draws from $D_n(e, d)$ and let θ_{n+1} be the solution to the optimization problem in line 10 of the n^{th} epoch of* EPOCHADEL*. Then with probability at least $1 - \delta$ we have:*

$$\mathbb{E}_{s,d \sim D_n}\left[\|D_n(a \mid s, d) - \mathbb{P}_{\pi_{\theta_{n+1}}}(a \mid s, d)\|_1\right] \leq C\sqrt{\frac{1}{m} \ln {|\Theta|}/{\delta}}, \tag{11}$$

where $C > 0$ is a universal constant.

Please see Agarwal et al. (2020) for a proof. Lemma 4 implies that assuming realizability, as $M \to \infty$, our learned solution converges to the Bayes optimal model pointwise on the support over $D_n(s, d)$. Since we are only interested in consistency, we will assume $M \to \infty$ and assume $\mathbb{P}_{n+1}(a \mid s, d) = D_n(a \mid s, d)$ for every $(s, d) \in \mathtt{supp} D_n(s, d)$. We will refer to this as optimally performing the maximum likelihood estimation at n^{th} epoch. If the learned policy is given by $\mathbb{P}_{n+1}(a \mid s, d) = D_n(a \mid s, d)$, then the next Lemma states the relationship between the marginal distribution $\mathbb{P}_{n+1}(a \mid s)$ for the next time epoch and marginal $\mathbb{P}_n(a \mid s)$ for this epoch.

Lemma 5 (Inductive Relation Between Marginals)**.** *For any $n \in \mathbb{N}$, if we optimally perform the maximum likelihood estimation at the n^{th} epoch of* EPOCHADEL*, then for all start states s, the marginal distribution $\mathbb{P}_{n+1}(a \mid s)$ for the $(n + 1)^{th}$ epoch is given by:*

$$\mathbb{P}_{n+1}(a \mid s) = \sum_d \frac{\mathbb{P}^\star(d \mid [s, a])\mathbb{P}_n(a \mid s)\mathbb{P}^\star(d \mid s)}{\sum_{a' \in \mathcal{A}} \mathbb{P}^\star(d \mid [s, a'])\mathbb{P}_n(a' \mid s)}.$$

Proof. The proof is completed as follows:

$$\mathbb{P}_{n+1}(a \mid s) = \sum_d \mathbb{P}^\star(d \mid s)\mathbb{P}_{n+1}(a \mid s, d) = \sum_d \frac{\mathbb{P}^\star(d \mid [s,a])\mathbb{P}_n(a \mid s)\mathbb{P}^\star(d \mid s)}{\sum_{a' \in \mathcal{A}} \mathbb{P}^\star(d \mid [s,a'])\mathbb{P}_n(a' \mid s)},$$

where the first step uses Lemma 2 and the second step uses $\mathbb{P}_{n+1}(a \mid s, d) = D_n(a \mid s, d)$ (optimally solving maximum likelihood) and the form of D_n from Lemma 3. □

A.1. Proof of Convergence for Marginal Distribution

Our previous analysis associates a probability distribution $\mathbb{P}_n(a \mid s, d)$ and $\mathbb{P}_n(a \mid s)$ with the n^{th} epoch of EPOCHADEL. For any $n \in \mathbb{N}$, the n^{th} epoch of EPOCHADEL can be viewed as a transformation of $\mathbb{P}_n(a \mid s, d) \mapsto \mathbb{P}_{n+1}(a \mid s, d)$ and $\mathbb{P}_n(a \mid s) \mapsto \mathbb{P}_{n+1}(a \mid s)$. In this section, we show that under certain conditions, the running average of the marginal distributions $\mathbb{P}_n(a \mid d)$ converges to the optimal marginal distribution $\mathbb{P}^\star(a \mid d)$. We then discuss how this can be used to learn the optimal policy $\mathbb{P}^\star(a \mid s, d)$.

We use a potential function approach to measure the progress of each epoch. Specifically, we will use KL-divergence as our choice of potential function. The next lemma bounds the change in potential after a single iteration.

Lemma 6. *[Potential Difference Lemma] For any $n \in \mathbb{N}$ and start state s, we define the following distribution over descriptions $\mathbb{P}_n(d \mid s) := \sum_{a' \in \mathcal{A}} \mathbb{P}^\star(d \mid [s,a])\mathbb{P}_n(a \mid s)$. Then for every start state s we have:*

$$\mathrm{D_{KL}}(\mathbb{P}^\star(a \mid s) \mid\mid \mathbb{P}_{n+1}(a \mid s)) - \mathrm{D_{KL}}(\mathbb{P}^\star(a \mid s) \mid\mid \mathbb{P}_n(a \mid s)) \le -\mathrm{D_{KL}}(\mathbb{P}^\star(d \mid s) \mid\mid \mathbb{P}_n(d \mid s)).$$

Proof. The change in potential from the start of n^{th} epoch to its end is given by:

$$\mathrm{D_{KL}}(\mathbb{P}^\star(a \mid s) \mid\mid \mathbb{P}_{n+1}(a \mid s)) - \mathrm{D_{KL}}(\mathbb{P}^\star(a \mid s) \mid\mid \mathbb{P}_n(a \mid s)) = -\sum_{a \in \mathcal{A}} \mathbb{P}^\star(a \mid s) \ln\left(\frac{\mathbb{P}_{n+1}(a \mid s)}{\mathbb{P}_n(a \mid s)}\right) \tag{12}$$

Using Lemma 5 and the definition of $\mathbb{P}_n(d \mid s)$ we get:

$$\frac{\mathbb{P}_{n+1}(a \mid s)}{\mathbb{P}_n(a \mid s)} = \sum_d \frac{\mathbb{P}^\star(d \mid [s,a])\mathbb{P}^\star(d \mid s)}{\sum_{a' \in \mathcal{A}} \mathbb{P}^\star(d \mid [s,a'])\mathbb{P}_n(a' \mid s)} = \sum_d \mathbb{P}^\star(d \mid [s,a])\frac{\mathbb{P}^\star(d \mid s)}{\mathbb{P}_n(d \mid s)}.$$

Taking logarithms and applying Jensen's inequality gives:

$$\ln\left(\frac{\mathbb{P}_{n+1}(a \mid s)}{\mathbb{P}_n(a \mid s)}\right) = \ln\left(\sum_d \mathbb{P}^\star(d \mid [s,a])\frac{\mathbb{P}^\star(d \mid s)}{\mathbb{P}_n(d \mid s)}\right) \ge \sum_d \mathbb{P}^\star(d \mid [s,a]) \ln\left(\frac{\mathbb{P}^\star(d \mid s)}{\mathbb{P}_n(d \mid s)}\right). \tag{13}$$

Taking expectations of both sides with respect to $\mathbb{P}^\star(a \mid s)$ gives us:

$$\begin{aligned}
\sum_a \mathbb{P}^\star(a \mid s) \ln\left(\frac{\mathbb{P}_{n+1}(a \mid s)}{\mathbb{P}_n(a \mid s)}\right) &\ge \sum_a \sum_d \mathbb{P}^\star(a \mid s)\mathbb{P}^\star(d \mid [s,a]) \ln\left(\frac{\mathbb{P}^\star(d \mid s)}{\mathbb{P}_n(d \mid s)}\right) \\
&= \sum_d \mathbb{P}^\star(d \mid s) \ln\left(\frac{\mathbb{P}^\star(d \mid s)}{\mathbb{P}_n(d \mid s)}\right) \\
&= \mathrm{D_{KL}}(\mathbb{P}^\star(d \mid s) \mid\mid \mathbb{P}_n(d \mid s))
\end{aligned}$$

where the last step uses the definition of $\mathbb{P}_n(d \mid s)$. The proof is completed by combining the above result with Equation 12. □

The $\mathbb{P}_s$ matrix. For a fixed start state s, we define $\mathbb{P}_s$ as the matrix whose entries are $\mathbb{P}^\star(d \mid [s,a])$. The columns of this matrix range over actions, and the rows range over descriptions. We denote the minimum singular value of the description matrix $\mathbb{P}_s$ by $\sigma_{min}(s)$.

We state our next assumption that the minimum singular value of $\mathbb{P}_s$ matrix is non-zero.

Assumption 2 (Minimum Singular Value is Non-Zero)**.** *For every start state s, we assume $\sigma_{min}(s) > 0$.*

Intuitively, this assumption states that there is enough information in the descriptions for the agent to decipher probabilities over actions from learning probabilities over descriptions. More formally, we are trying to decipher $\mathbb{P}^\star(a \mid s)$ using access to two distributions: $\mathbb{P}^\star(d \mid s)$ which generates the initial requests, and the teacher model $\mathbb{P}^\star(d \mid [s,a])$ which is used to describe an execution $e = [s,a]$. This can result in an underspecified problem. The only constraints these two distributions place on $\mathbb{P}^\star(a \mid s)$ is that $\sum_{a \in \mathcal{A}} \mathbb{P}^\star(d \mid [s,a])\mathbb{P}^\star(a \mid s) = \mathbb{P}^\star(d \mid s)$. This means all we know is that $\mathbb{P}^\star(a \mid s)$ belongs to the following set of solutions of the previous linear systems of equation:

$$\left\{ Q(a \mid s) \mid \sum_{a \in \mathcal{A}} \mathbb{P}^\star(d \mid [s,a]) Q(a \mid s) = \mathbb{P}^\star(d \mid s)\ \forall d, \qquad Q(a \mid s) \text{ is a distribution} \right\}.$$

As $\mathbb{P}^\star(a \mid s)$ belongs to this set hence this set is nonempty. However, if we also assume that $\sigma_{min}(s) > 0$ then the above set has a unique solution. Recall that singular values are square root of eigenvalues of $\mathbb{P}_s^\top \mathbb{P}_s$, and so $\sigma_{min}(s) > 0$ implies that the matrix $\mathbb{P}_s^\top \mathbb{P}_s$ is invertible. [7] This means, we can find the unique solution of the linear systems of equation by multiplying both sides by $(\mathbb{P}_s^\top \mathbb{P}_s)^{-1}\mathbb{P}_s^\top$. Hence, Assumption 2 makes it possible for us to find $\mathbb{P}^\star(a \mid s)$ using just the information we have. Note that we cannot solve the linear system of equations directly since the description space and action space can be extremely large. Hence, we use an oracle based solution via reduction to supervised learning.

The next theorem shows that the running average of learned probabilities $\mathbb{P}_n(a \mid s)$ converges to the optimal marginal distribution $\mathbb{P}^\star(a \mid s)$ at a rate determined by the inverse square root of the number of epochs of ADEL, the minimum singular value of the matrix $\mathbb{P}_s$, and the KL-divergence between optimal marginal and initial value.

Theorem 7. *[Rate of Convergence for Marginal] For any $t \in \mathbb{N}$ we have:*

$$\|\mathbb{P}^\star(a \mid s) - \frac{1}{t}\sum_{n=1}^{t} \mathbb{P}_n(a \mid s)\|_2 \le \frac{1}{\sigma_{min}(s)} \sqrt{\frac{2}{t} \mathrm{D}_{\mathrm{KL}}(\mathbb{P}^\star(a \mid s) \mid\mid \mathbb{P}_1(a \mid s))},$$

and if $\mathbb{P}_1(a \mid s, d)$ is a uniform distribution for every s and d, then

$$\|\mathbb{P}^\star(a \mid s) - \frac{1}{t}\sum_{n=1}^{t} \mathbb{P}_n(a \mid s)\|_2 \le \frac{1}{\sigma_{min}(s)} \sqrt{\frac{2\ln|\mathcal{A}|}{t}}.$$

Proof. We start with Lemma 6 and bound the right hand side as shown:

$$\begin{aligned}
\mathrm{D}_{\mathrm{KL}}(\mathbb{P}^\star(a \mid s) \mid\mid \mathbb{P}_{n+1}(a \mid s)) - \mathrm{D}_{\mathrm{KL}}(\mathbb{P}^\star(a \mid s) \mid\mid \mathbb{P}_n(a \mid s)) &\le -\mathrm{D}_{\mathrm{KL}}(\mathbb{P}^\star(d \mid s) \mid\mid \mathbb{P}_n(d \mid s)) \\
&\le -\frac{1}{2}\|\mathbb{P}^\star(d \mid s) - \mathbb{P}_n(d \mid s)\|_1^2, \\
&\le -\frac{1}{2}\|\mathbb{P}^\star(d \mid s) - \mathbb{P}_n(d \mid s)^2\|_2^2 \\
&= -\frac{1}{2}\|\mathbb{P}_s \left\{\mathbb{P}^\star(a \mid s) - \mathbb{P}_n(a \mid s)\right\}\|_2^2, \\
&\le -\frac{1}{2}\sigma_{min}(s)^2 \|\mathbb{P}^\star(a \mid s) - \mathbb{P}_n(a \mid s)\|_2^2,
\end{aligned}$$

where the second step uses Pinsker's inequality. The third step uses the property of p-norms, specifically, $\|\nu\|_2 \le \|\nu\|_1$ for all ν. The fourth step, uses the definition of $\mathbb{P}^\star(d \mid s) = \sum_{a' \in \mathcal{A}} \mathbb{P}(d \mid s, a')\mathbb{P}^\star(a' \mid s))$ and $\mathbb{P}_n(d \mid s) = \sum_{a' \in \mathcal{A}} \mathbb{P}(d \mid s, a')\mathbb{P}_n(a' \mid s)$. We interpret the notation $\mathbb{P}^\star(a \mid s)$ as a vector over actions whose value is the probability $\mathbb{P}^\star(a \mid s)$. Therefore, $\mathbb{P}_s \mathbb{P}^\star(a \mid s)$ represents a matrix-vector multiplication. Finally, the last step, uses $\|Ax\|_2 \ge \sigma_{min}(A)\|x\|_2$ for any vector x and matrix A of compatible shape such that Ax is defined, where $\sigma_{min}(A)$ is the smallest singular value of A.

Summing over n from $n = 1$ to t and rearranging the terms we get:

$$\mathrm{D}_{\mathrm{KL}}(\mathbb{P}^\star(a \mid s) \mid\mid \mathbb{P}_{t+1}(a \mid s)) \le \mathrm{D}_{\mathrm{KL}}(\mathbb{P}^\star(a \mid s) \mid\mid \mathbb{P}_1(a \mid s)) - \frac{1}{2}\sigma_{min}(s)^2 \sum_{n=1}^{t} \|\mathbb{P}^\star(a \mid s) - \mathbb{P}_n(a \mid s)\|_2^2.$$

[7] Recall that a matrix of the form $A^\top A$ always have non-negative eigenvalues.

As the left hand-side is positive we get:

$$\sum_{n=1}^{t} \|\mathbb{P}^{\star}(a \mid s) - \mathbb{P}_n(a \mid s)\|_2^2 \leq \frac{2}{\sigma_{min}(s)^2} \mathrm{D}_{\mathrm{KL}}(\mathbb{P}^{\star}(a \mid s) \mid\mid \mathbb{P}_1(a \mid s)).$$

Dividing by t and applying Jensen's inequality (specifically, $\mathbb{E}[X^2] \geq \mathbb{E}[|X|]^2$) we get:

$$\frac{1}{t} \sum_{n=1}^{t} \|\mathbb{P}^{\star}(e) - \mathbb{P}_n(e)\|_2 \leq \frac{1}{\sigma_{min}(s)} \sqrt{\frac{2}{t} \mathrm{D}_{\mathrm{KL}}(\mathbb{P}^{\star}(a \mid s) \mid\mid \mathbb{P}_1(a \mid s))} \tag{14}$$

Using the triangle inequality, the left hand side can be bounded as:

$$\frac{1}{t} \sum_{n=1}^{t} \|\mathbb{P}^{\star}(a \mid s) - \mathbb{P}_n(a \mid s)\|_2 \geq \|\mathbb{P}^{\star}(a \mid s) - \frac{1}{t} \sum_{n=1}^{t} \mathbb{P}_n(a \mid s)\|_2 \tag{15}$$

Combining the previous two equations proves the main result. Finally, note that if $\mathbb{P}_1(a \mid s, d) = {}^1/_{|\mathcal{A}|}$ for every value of s, d, and a, then $\mathbb{P}_1(a \mid s)$ is also a uniform distribution over actions. The initial KL-divergence is then bounded by $\ln |\mathcal{A}|$ as shown below:

$$\mathrm{D}_{\mathrm{KL}}(\mathbb{P}^{\star}(a \mid s) \mid\mid \mathbb{P}_1(a \mid s)) = -\sum_{a \in \mathcal{A}} \mathbb{P}^{\star}(a \mid s) \ln \frac{1}{|\mathcal{A}|} + \sum_{a \in \mathcal{A}} \mathbb{P}^{\star}(a \mid s) \ln \mathbb{P}^{\star}(a \mid s) \leq \ln |\mathcal{A}|,$$

where the second step uses the fact that entropy of a distribution is non-negative. This completes the proof. □

A.2. Proof of Convergence to Near-Optimal Policy

Finally, we discuss how to learn $\mathbb{P}^{\star}(a \mid s, d)$ once we learn $\mathbb{P}^{\star}(a \mid s)$. Since we only derive convergence of running average of $\mathbb{P}_n(a \mid s)$ to $\mathbb{P}^{\star}(a \mid s)$, therefore, we cannot expect $\mathbb{P}_n(a \mid s, d)$ to converge to $\mathbb{P}^{\star}(a \mid s, d)$. Instead, we will show that if we perform line 4-10 in Algorithm 4 using the running average of policies, then the learned Bayes optimal policy will converge to the near-optimal policy. The simplest way to accomplish this with Algorithm 4 is to perform the block of code in line 4-10 twice, once when taking actions according to $\mathbb{P}_n(a \mid s, d)$, and once when taking actions according to running average policy $\tilde{\mathbb{P}}_n(a \mid s, d) = \frac{1}{n} \sum_{t=1}^{n} \tilde{\mathbb{P}}_t(a \mid s, d)$. This will give us two Bayes optimal policy in 10 one each for the current policy $\mathbb{P}_n(a \mid s, d)$ and the running average policy $\tilde{\mathbb{P}_n}(a \mid s, d)$. We use the former for roll-in in the future and the latter for evaluation on held-out test set.

For convenience, we first define an operator that denotes mapping of one agent policy to another.

W operator. Let $\mathbb{P}(a \mid s, d)$ be an agent policy used to generate data in any epoch of EPOCHADEL (line 5-9). We define the W operator as the mapping to the Bayes optimal policy for the optimization problem solved by EPOCHADEL in line 10 which we denote by $(W\mathbb{P})$. Under the realizability assumption (Assumption 1), the agent learns the $W\mathbb{P}$ policy when $M \to \infty$. Using Lemma 2 and Lemma 3, we can verify that:

$$(W\mathbb{P})(a \mid s, d) = \frac{\mathbb{P}^{\star}(d \mid [s, a])\mathbb{P}(a \mid s)}{\sum_{a' \in \mathcal{A}} \mathbb{P}^{\star}(d \mid [s, a'])\mathbb{P}(a' \mid s)}, \quad \text{where} \quad \mathbb{P}(a \mid s) = \sum_{d} \mathbb{P}^{\star}(d \mid s)\mathbb{P}(a \mid s, d).$$

We first show that our operator is smooth around $\mathbb{P}^{\star}(a \mid s)$.

Lemma 8 (Smoothness of W)**.** *For any start state s and description $d \in \texttt{supp}\, \mathbb{P}^{\star}(d \mid s)$, there exists a finite constant K_s such that:*

$$\|W\mathbb{P}(a \mid s, d) - W\mathbb{P}^{\star}(a \mid s, d)\|_1 \leq K_s \|\mathbb{P}(a \mid s) - \mathbb{P}^{\star}(a \mid s)\|_1.$$

Proof. We define $\mathbb{P}(d \mid s) = \sum_{a' \in \mathcal{A}} \mathbb{P}^\star(d \mid s, a')\mathbb{P}(a' \mid s)$. Then from the definition of operator W we have:

$$\begin{aligned}
&|W\mathbb{P}(a \mid s, d) - W\mathbb{P}^\star(a \mid s, d)|_1 \\
&= \sum_{a \in \mathcal{A}} \left| \frac{\mathbb{P}^\star(d \mid [s,a])\mathbb{P}(a \mid s)}{\mathbb{P}(d \mid s)} - \frac{\mathbb{P}^\star(d \mid [s,a])\mathbb{P}^\star(a \mid s)}{\mathbb{P}^\star(d \mid s)} \right| \\
&= \sum_{a \in \mathcal{A}} \mathbb{P}^\star(d \mid [s,a]) \frac{|\mathbb{P}(a \mid s)\mathbb{P}^\star(d \mid s) - \mathbb{P}^\star(a \mid s)\mathbb{P}(d \mid s)|}{\mathbb{P}(d \mid s)\mathbb{P}^\star(d \mid s)} \\
&\leq \sum_{a \in \mathcal{A}} \mathbb{P}^\star(d \mid [s,a])\mathbb{P}(a \mid s) \frac{|\mathbb{P}^\star(d \mid s) - \mathbb{P}(d \mid s)|}{\mathbb{P}(d \mid s)\mathbb{P}^\star(d \mid s)} + \sum_{a \in \mathcal{A}} \mathbb{P}^\star(d \mid [s,a]) \frac{|\mathbb{P}(a \mid s) - \mathbb{P}^\star(a \mid s)|}{\mathbb{P}^\star(d \mid s)} \\
&= \frac{|\mathbb{P}^\star(d \mid s) - \mathbb{P}(d \mid s)|}{\mathbb{P}^\star(d \mid s)} + \sum_{a \in \mathcal{A}} \mathbb{P}^\star(d \mid [s,a]) \frac{|\mathbb{P}(a \mid s) - \mathbb{P}^\star(a \mid s)|}{\mathbb{P}^\star(d \mid s)} \\
&\leq 2 \sum_{a \in \mathcal{A}} \mathbb{P}^\star(d \mid [s,a]) \frac{|\mathbb{P}(a \mid s) - \mathbb{P}^\star(a \mid s)|}{\mathbb{P}^\star(d \mid s)}, \qquad \text{(using the definition of } \mathbb{P}(d \mid s)\text{)} \\
&\leq \frac{2}{\mathbb{P}^\star(d \mid s)} \|\mathbb{P}(a \mid s) - \mathbb{P}^\star(a \mid s)\|_1.
\end{aligned}$$

Note that the policy will only be called on a given pair of (s, d) if and only if $\mathbb{P}^\star(d \mid s) > 0$, hence, the constant is bounded. We define $K_s = \max_d \frac{2}{\mathbb{P}^\star(d|s)}$ where maximum is taken over all descriptions $d \in \texttt{supp}\, \mathbb{P}^\star(d \mid s)$. □

Theorem 9 (Convergence to Near Optimal Policy)**.** *Fix $t \in \mathbb{N}$, and let $\tilde{\mathbb{P}}_t(a \mid s, d) = \frac{1}{t}\sum_{n=1}^t \mathbb{P}_n(a \mid s, d)$ be the average of the agent's policy across epochs. Then for every start state s and description $d \in \texttt{supp}\, \mathbb{P}^\star(d \mid s)$ we have:*

$$\lim_{t \to \infty} (W\tilde{\mathbb{P}}_t)(a \mid s, d) = \mathbb{P}^\star(a \mid s, d).$$

Proof. Let $\tilde{\mathbb{P}}_t(a \mid s) = \sum_d \mathbb{P}^\star(d \mid s)\tilde{\mathbb{P}}_t(a \mid s, d)$. Then it is easy to see that $\tilde{\mathbb{P}}_t(a \mid s) = \frac{1}{t}\sum_{n=1}^t \mathbb{P}_n(a \mid s)$. From Theorem 7 we have $\lim_{t\to\infty} \|\tilde{\mathbb{P}}_t(a \mid s) - \mathbb{P}^\star(a \mid s)\|_2 = 0$. As $\mathcal{A}$ is finite dimensional, therefore, $\|\cdot\|_2$ and $\|\cdot\|_1$ are equivalent, i.e., convergence in one also implies convergence in the other. This implies, $\lim_{t\to\infty} \|\tilde{\mathbb{P}}_t(a \mid s) - \mathbb{P}^\star(a \mid s)\|_1 = 0$.

From Lemma 8 we have:

$$\lim_{t \to \infty} \|(W\tilde{\mathbb{P}}_t)(a \mid s, d) - (W\mathbb{P}^\star)(a \mid s, d)\|_1 \leq K_s \lim_{t \to \infty} \|\tilde{\mathbb{P}}_t(a \mid s) - \mathbb{P}^\star(a \mid s)\|_1 = 0.$$

This shows $\lim_{t\to\infty}(W\tilde{\mathbb{P}}_t)(a \mid s, d) = (W\mathbb{P}^\star)(a \mid s, d)$. Lastly, we show that the optimal policy $\mathbb{P}^\star(a \mid s, d)$ is a fixed point of W:

$$(W\mathbb{P}^\star)(a \mid s, d) = \frac{\mathbb{P}^\star(d \mid s, a)\mathbb{P}^\star(a \mid s)}{\sum_{a' \in \mathcal{A}} \mathbb{P}^\star(d \mid s, a')\mathbb{P}^\star(a' \mid s)} = \frac{\mathbb{P}^\star(d, a \mid s)}{\sum_{a' \in \mathcal{A}} \mathbb{P}^\star(d, a' \mid s)} = \frac{\mathbb{P}^\star(d, a \mid s)}{\mathbb{P}^\star(d \mid s)} = \mathbb{P}^\star(a \mid s, d).$$

This completes the proof. □

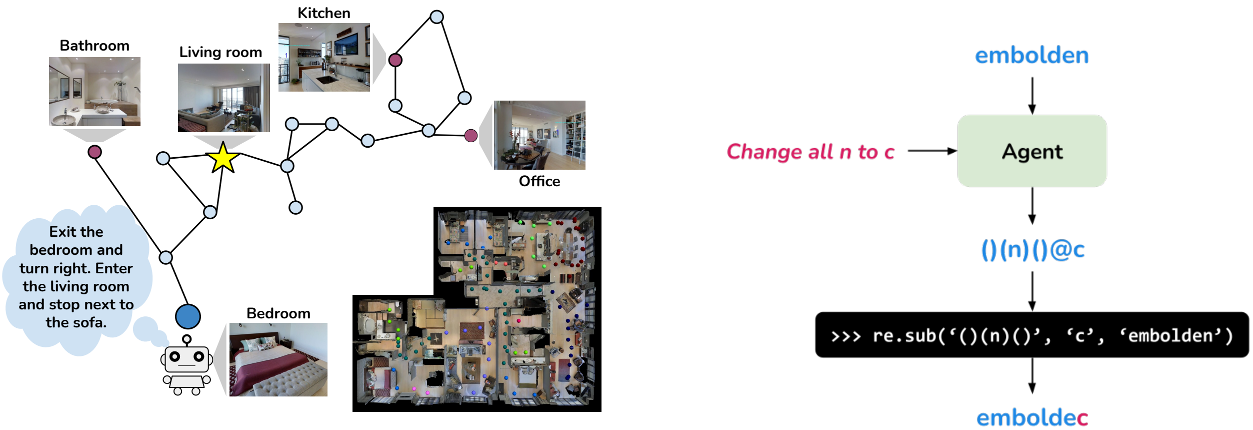
B. Problem settings

Figure 3 illustrates the two problems that we conduct experiments on.

B.1. Vision-Language Navigation

Environment Simulator and Data. We use the Matterport3D simulator and the Room-to-Room dataset[8] developed by Anderson et al. (2018). The simulator photo-realistically emulates the first-person view of a person walking in a house. The dataset contains tuples of human-generated English navigation requests annotated with ground-truth paths in the

[8] *https://github.com/peteanderson80/Matterport3DSimulator/blob/master/tasks/R2R/data/download.sh*



(a) *Vision-language navigation* (NAV): a (robot) agent fulfills a navigational natural-language request in a photo-realistic simulated house. Locations in the house are connected as a graph. In each time step, the agent receives a photo of the panoramic view at its current location (due to space limit, here we only show part of a view). Given the view and the language request, the agent chooses an adjacent location to go to. On average, each house has about 117 locations.

(b) *Word modification* (REGEX): an agent is given an input word and a natural-language request that asks it to modify the word. The agent outputs a regular expression that follows our specific syntax. The regular expression is executed by the Python's `re.sub()` method to generate an output word.

Figure 3. Illustrations of the two request-fulfilling problems that we conduct experiments on.

environments. To evaluate on the test set, the authors require submitting predictions to an evaluation site[9], which limits the number of submissions to five. As our goal is not to establish state-of-the-art results on this task, but to compare performance of multiple learning frameworks, we re-split the data into 4,315 simulation, 2,100 validation, and 2,349 test data points. The simulation split, which is used to simulate the teacher, contains three requests per data point (i.e. $|\mathcal{D}_n^\star| = 3$). The validation and test splits each contains only one request per data point. On average, each request includes 2.5 sentences and 26 words. The word vocabulary size is 904 and the average number of optimal actions required to reach the goal is 6.

Simulated Teacher. We use SDTW (Magalhaes et al., 2019) as the `perf` metric and set the threshold $\tau = 0.5$. The SDTW metric re-weights success rate by the shortest (order-preserving) alignment distance between a predicted path and a ground-truth path, offering more fine-grained evaluation of navigation paths.

Approximate marginal $P_{\pi_\omega}(e \mid s_1)$. The approximate marginal is a function that takes in a start location s_1 and randomly samples a shortest path on the environment graph that starts at s_1 and has (unweighted) length between 2 and 6.

B.2. Word Modification

Regular Expression Compiler. We use Python 3.7's `re.sub(pattern, replace, string)` method as the regular expression compiler. The method replaces every substring of `string` that matches a regular expression `pattern` with the string `replace`. A regular expression predicted by our agent $\hat{a}_{1:H}$ has the form "`pattern@replace`", where `pattern` and `replace` are strings and `@` is the at-sign character. For example, given the word *embolden* and the request "*replace all n with c*", the agent should ideally generate the regular expression "`()(n)()@c`". We then split the regular expression by the character `@` into a string `pattern` = "`()(n)()`" and a string `replace` = "*c*". We execute the Python's command `re.sub('()(n)()', 'c', 'embolden')` to obtain the output word *emboldec*.

Data. We use the data collected by Andreas et al. (2018). The authors presented crowd-workers with pairs of input and output words where the output words are generated by applying regular expressions onto the input words. Workers are asked to write English requests that describe the change from the input words to the output words. From the human-generated requests, the authors extracted 1,917 request templates. For example, a template has the form *add an* `AFTER` *to the start of words beginning with* `BEFORE`, where `AFTER` and `BEFORE` can be replaced with latin characters to form a request. Each request template is annotated with a regular expression template that it describes. Since the original dataset is not designed to

[9] *https://eval.ai/web/challenges/challenge-page/97/overview*

Algorithm 5 ADEL: Learning from Activity Describers via Semi-Supervised Exploration (experimental version).

Input: teacher model $\mathbb{P}_T(d \mid e)$, approximate marginal $\mathbb{P}_{\pi_\omega}(e \mid s_1)$, mixing weight $\lambda \in [0, 1]$
Initialize policy $\pi_\theta : \mathcal{S} \times \mathcal{D} \to \Delta(\mathcal{A})$
Initialize policy $\pi_\beta : \mathcal{S} \times \mathcal{D} \to \Delta(\mathcal{A})$
for $n = 1, 2, \cdots, N$ **do**
  Word samples $q = (R, s_1, d^\star) \sim \mathbb{P}^\star(\cdot)$
  Agent generates $\hat{e} \sim \mathbb{P}_{\pi_\beta}(\cdot \mid s_1, d^\star)$
  Teacher generates $\hat{d} \sim \mathbb{P}_T(\cdot \mid \hat{e})$
  Agent samples $\tilde{e} \sim \mathbb{P}_{\pi_\omega}(\cdot \mid s_1)$
  Compute losses:

$$\mathcal{L}(\theta) = \sum_{(s, \hat{a}_s) \in \hat{e}} \log \pi_\theta(\hat{a}_s \mid s, \hat{d})$$

$$\mathcal{L}(\beta) = \lambda \sum_{(s, \tilde{a}_s) \in \tilde{e}} \log \pi_\beta(\tilde{a}_s \mid s, \hat{d}) + (1 - \lambda) \sum_{(s, \hat{a}_s) \in \hat{e}} \log \pi_\beta(\hat{a}_s \mid s, \hat{d})$$

  Compute gradients $\nabla \mathcal{L}(\theta)$ and $\nabla \mathcal{L}(\beta)$
  Use gradient descent to update θ and β with $\nabla \mathcal{L}(\theta)$ and $\nabla \mathcal{L}(\beta)$, respectively
return $\pi : s, d \mapsto \mathrm{argmax}_a \, \pi_\theta(a \mid s, d)$

evaluate generalization to previously unseen request templates, we modified the script provided by the authors to generate a new dataset where the simulation and evaluation requests are generated from disjoint sets of request templates. We select 110 regular expressions templates that are each annotated with more than one request template. Then, we further remove pairs of regular expression and request templates that are mistakenly paired. We end up with 1111 request templates describing these 110 regular expression templates. We use these templates to generate tuples of requests and regular expressions. In the end, our dataset consists of 114,503 simulation, 6,429 validation, and 6,429 test data points. The request templates in the simulation, validation, and test sets are disjoint.

Simulated Teacher. We extend the performance metric `perf` in §4.1 to evaluating multiple executions. Concretely, given executions $\{w_j^{\text{inp}}, \hat{w}_j^{\text{out}}\}_{j=1}^J$, the metric counts how many pairs where the predicted output word matches the ground-truth: $\sum_{j=1}^J \mathbb{1}\left\{\hat{w}_j^{\text{out}} = w_j^{\text{out}}\right\}$. We set the threshold $\tau = J$.

Approximate marginal $P_{\pi_\omega}(e \mid s_1)$. The approximate marginal is a uniform distribution over a dataset of (unlabeled) regular expressions. These regular expressions are generated using the code provided by Andreas et al. (2018).[10]

C. Practical Implementation of ADEL

In our experiments, we employ the following implementation of ADEL (Alg 5), which learns a policy π_β such that $\mathbb{P}_{\pi_\beta}(e \mid s_1, d)$ approximates the mixture $\tilde{\mathbb{P}}(e \mid s_1, d)$ in Alg 3. In each episode, we sample an execution $\hat{e}$ using the policy π_β. Then, similar to Alg 3, we ask the teacher $\mathbb{P}_T$ for a description of $\hat{e}$ and the use the pair $(\hat{e}, \hat{d})$ to update the agent policy π_θ. To ensure that $\mathbb{P}_{\pi_\beta}$ approximates $\tilde{\mathbb{P}}$, we draw a sample $\tilde{e}$ from the approximate marginal $\mathbb{P}_{\pi_\omega}(e \mid s_1)$ and update π_β using a λ-weighted loss of the log-likelihoods of the two data points $(\tilde{e}, \hat{d})$ and $(\hat{e}, \hat{d})$. We only use $(\hat{e}, \hat{d})$ to update the agent policy π_θ.

An alternative (naive) implementation of sampling from the mixture $\tilde{\mathbb{P}}$ is to first choose a policy between π_ω (with probability λ) and π_θ (with probability $1 - \lambda$), and then use this policy to generate an execution. Compared to this approach, our implementation has two advantages:

1. Sampling from the mixture is simpler: instead of choosing between π_θ and π_ω, we always use π_β to generate executions;

2. More importantly, samples are more diverse: in the naive approach, the samples are either completely request-agnostic

[10] *https://github.com/jacobandreas/l3/blob/master/data/re2/generate.py*

	NAV		REGEX	
Anneal λ every L steps	Success rate (%) ↑	Sample complexity ↓	Success rate (%) ↑	Sample complexity ↓
L = 2000	31.4	304K	87.7	368K
L = 5000	32.5	384K	86.4	448K
No annealing (final)	32.0	384K	88.0	608K

Table 5. Effects of annealing the mixing weight λ. When annealed, the mixing weight is updated as $\lambda \leftarrow \max(\lambda_{\text{min}}, \lambda \cdot \beta)$, where the annealing rate $\beta = 0.5$ and the minimum mixing rate $\lambda_{\text{min}} = 0.1$. Initially, λ is set to be 0.5. All results are on validation data. Sample complexity is the number of training episodes required to reach a success rate of at least c ($c = 30\%$ in NAV, and $c = 85\%$ in REGEX).

(if generated by π_ω) or completely request-guided (if generated by π_θ). As a machine learning-based model that learns from a mixture of data generated by π_ω and π_θ, the policy π_β can generalize and generate executions that are partially request-agnostic.

Effects of the Annealing Mixing Weight. We do not anneal the mixing weight λ in our experiments. Table 5 shows the effects of annealing the mixing weight with various settings. We find that annealing improves the sample complexity of the agents, i.e. they reach a substantially high success rate in less training episodes. But overall, not annealing yields slightly higher final success rates.

D. Training details

Reinforcement learning's continuous reward. In REGEX, the continuous reward function is

$$\frac{|w^{\text{out}}| - \texttt{editdistance}\,(\hat{w}^{\text{out}}, w^{\text{out}})}{|w^{\text{out}}|} \tag{16}$$

where w^{out} is the ground-truth output word, $\hat{w}^{\text{out}}$ is the predicted output word, $\texttt{editdistance}(.,.)$ is the string edit distance computed by the Python's editdistance module.

In NAV, the continuous reward function is

$$\frac{\texttt{shortest}\,(s_1, s_g) - \texttt{shortest}\,(s_H, s_g)}{\texttt{shortest}\,(s_1, s_g)} \tag{17}$$

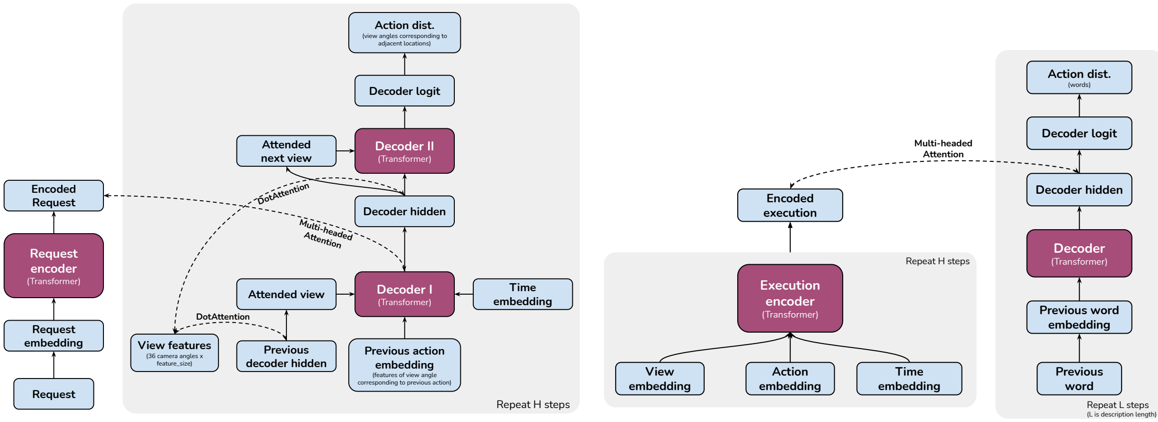
where s_1 is the start location, s_g is the goal location, s_H is the agent's final location, and $\texttt{shortest}(.,.)$ is the shortest-path distance between two locations (according to the environment's navigation graph).

Model architecture. Figure 4 and Figure 5 illustrate the architectures of the models that we train in the two problems, respectively. For each problem, we describe the architectures of the student policy $\pi_\theta(a \mid s, d)$ and the teacher's language model $\tilde{\mathbb{P}}(d \mid e)$. All models are encoder-decoder models but the NAV models use Transformer as the recurrent module while REGEX models use LSTM.

Hyperparameters. Model and training hyperparameters are provided in Table 6. Each model is trained on a single NVIDIA V100 GPU, GTX 1080, or Titan X. Training with the ADEL algorithm takes about 19 hours for NAV and 14 hours for REGEX on a machine with an Intel i7-4790K 4.00GHz CPU and a Titan X GPU.

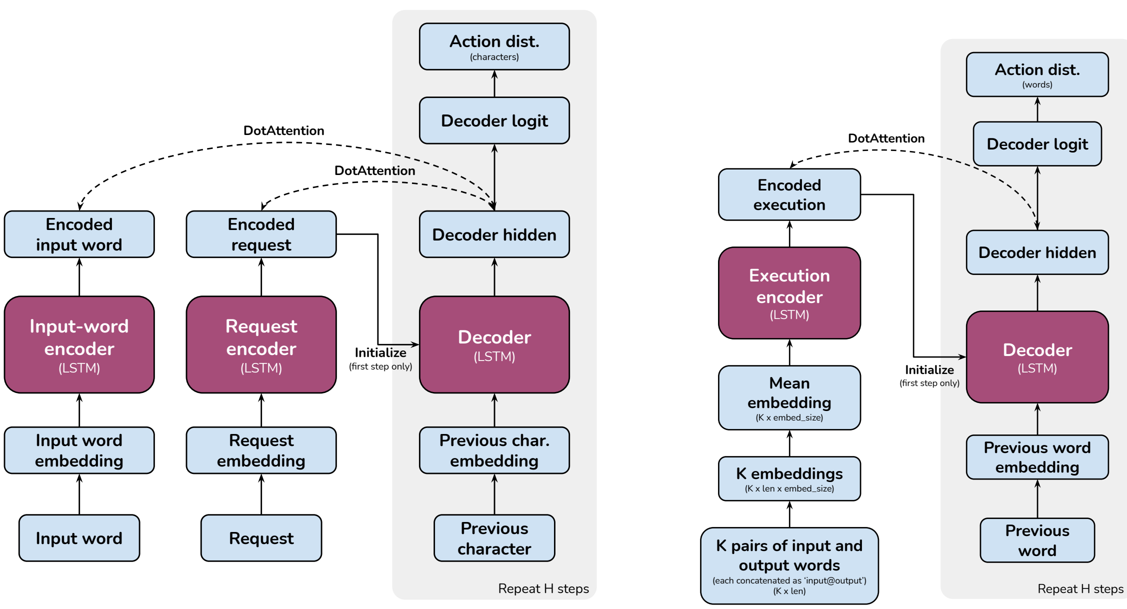
E. Qualitative examples

Figure 6 and Table 7 show the qualitative examples in the NAV and REGEX problems, respectively.



(a) Student model (b) Teacher model

Figure 4. Student and teacher models in NAV.



(a) Student model (b) Teacher model

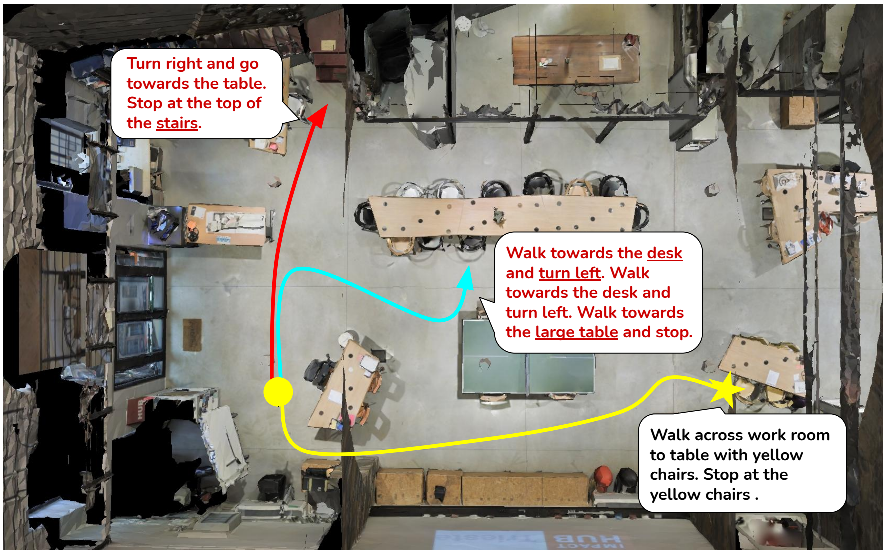
Figure 5. Student and teacher models in REGEX.

Hyperparameter	NAV	REGEX
Student policy π_θ and **Teacher's describer model** $\tilde{P}_T$		
Base architecture	Transformer	LSTM
Hidden size	256	512
Number of hidden layers (of each encoder or decoder)	1	1
Request word embedding size	256	128
Character embedding size (for the input and output words)	-	32
Time embedding size	256	-
Attention heads	8	1
Observation feature size	2048	-
Teacher simulation		
`perf` metric	STDW (Magalhaes et al., 2019)	Number of output words matching ground-truths
Number of samples for approximate pragmatic inference ($\lvert\mathcal{D}_{\text{cand}}\rvert$)	5	10
Threshold (τ)	0.5	$J = 5$
Training		
Time horizon (H)	10	40
Batch size	32	32
Learning rate	10^{-4}	10^{-3}
Optimizer	Adam	Adam
Number of training iterations	25K	30K
Mixing weight (λ, no annealing)	0.5	0.5

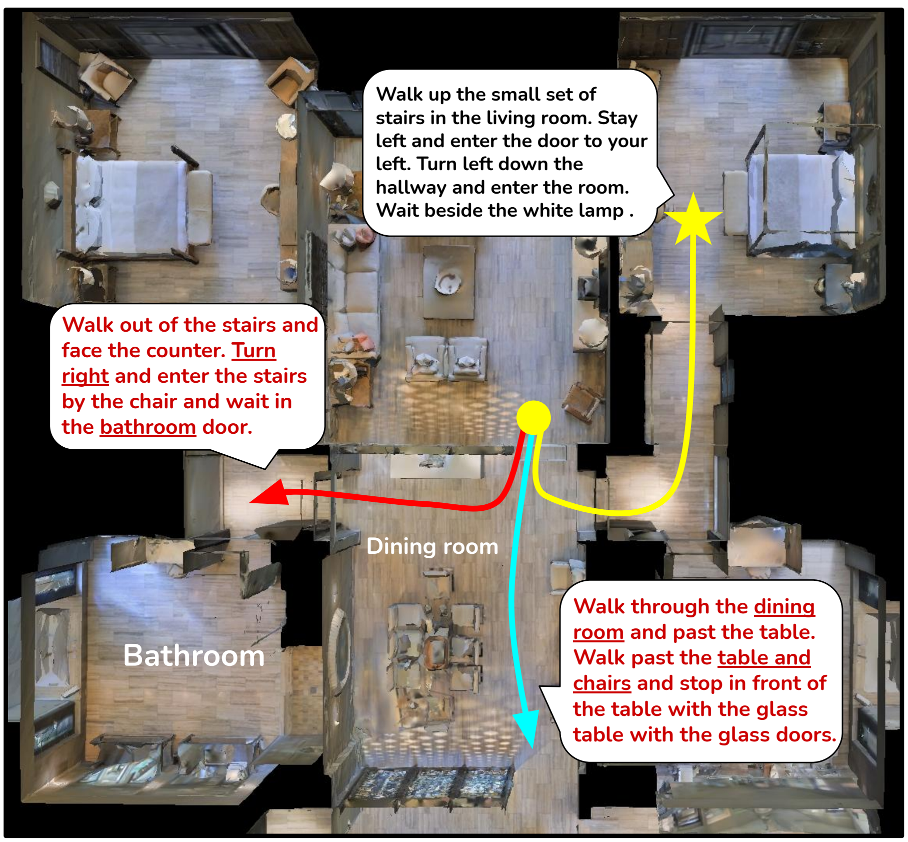
Table 6. Hyperparameters for training with the ADEL algorithm.

Input word	Output word	Description generated by $\tilde{\mathbb{P}}(d \mid e)$
attendant	xjtendxjt	replace [a] and the letter that follows it with an [x j]
disclaims	esclaims	if the word does not begin with a vowel , replace the first two letters with [e]
inculpating	incuxlpating	for any instance of [l] add a [x] before the [l]
flanneling	glanneling	change the first letter of the word to [g]
dhoti	jhoti	replaced beginning of word with [j]
stuccoing	ostuccoing	all words get a letter [o] put in front
reappearances	reappearanced	if the word ends with a consonant , change the consonant to [d]
bigots	vyivyovyvy	replace each consonant with a [v y]

Table 7. Qualitative examples in the REGEX problem. We show pairs of input and output words and how the teacher's language model $\tilde{\mathbb{P}}(d \mid e)$ describes the modifications applied to the input words.



(a)



(b)

Figure 6. Qualitative examples in the NAV problem. The **black texts** (no underlines) are the initial requests $d^\star$ generated by humans. The paths are the ground-truth paths implied by the requests. and are some paths are taken by the agent during training. Here, we only show two paths per example. The **red texts** are descriptions $\hat{d}$ generated by the teacher's learned (conditional) language model $\tilde{\mathbb{P}}(d \mid e)$. We show the bird-eye views of the environments for better visualization, but the agent only has access to the first-person panoramic views at its locations.