Table 4. List of common notations and their definitions.

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

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 \( \mathbb{P}_T(d \mid e) \).
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 $a_1 \in A$ 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 $D_n(s,d) = \sum_{a' \in A} D_n([s, a'], d)$. Secondly, let $P^*(s)$ be the marginal distribution over start state $s$ given by $E_{(R,d,s_1)} P^*(q) \{ \{ s_1 = s \} \}$. 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., $P_T(d | e) = P^*(d | e)$ for every $(d, e)$.

1: **Input**: teacher model $P^*(d | e)$ and task distribution model $P^*(q)$.
2: Initialize agent policy $\pi_{\theta_1} : S \times D \rightarrow \text{unf}(A)$
3: for $n = 1, 2, \cdots, N$ do
4: $\mathcal{B} = \emptyset$
5: for $m = 1, 2, \cdots, M$ do
6: World samples $q = (R, d^*, s_1) \sim P^*(\cdot)$
7: Agent generates $\hat{e} \sim P_{\pi_{\theta_n}}(\cdot | s_1, d^*)$
8: Teacher generates description $\hat{d} \sim P^*(\cdot | \hat{e})$
9: $\mathcal{B} \leftarrow \mathcal{B} \cup \{ (\hat{e}, \hat{d}) \}$
10: Update agent policy using batch updates:
   
   $\theta_{n+1} \leftarrow \arg\max_{\theta \in \Theta} \sum_{(e,d) \in \mathcal{B}} \sum_{(s,a) \in \hat{e}} \log \pi_{\theta}(a_s | s, \hat{d})$

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

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

$$P_n(e := [s, a]) = P^*(s) P_n(a | s), \text{ where } P_n(a | s) := \sum_d P^*(d | s) P_n(a | s, d).$$

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

$$\sum_{a' \in A} P_n(e' := [s, a']) = \sum_{a' \in A} \sum_{R,d} P^*(R,d,s) P_n(a' | s,d) = \sum_{R,d} P^*(R,d,s) = P^*(s).$$

Next we compute the conditional distribution $P_n(a | s)$ as shown:

$$P_n(a | s) = \frac{P_n([s, a])}{\sum_{a' \in A} P_n([s, a'])} = \frac{\sum_{R,d} P^*(R,d,s) P_n(a | s, d)}{P^*(s)} = \sum_d P^*(s,d) P_n(a | s, d) = \sum_d P^*(d | s) P_n(a | s, d).$$

This also proves $P_n([s, a]) = P^*(s) P_n(a | s)$. \qed
For $H = 1$, the update equation in line 10 solves the following optimization equation:

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

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

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

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 $a \in A$ and every $(s, d) \in \text{supp}D_n(s, d)$. 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 $(s, d) \in \text{supp}D_n(s, d)$ the value of the Bayes optimal model $D_n(a \mid s, d)$ at the end of the $n^{th}$ epoch is given by:

$$D_n(a \mid s, d) = \frac{P^*(d \mid [s, a])P_n(a \mid s)}{\sum_{a' \in A} P^*(d \mid [s, a'])P_n(a' \mid s)}.
$$

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

$$D_n(a \mid s, d) = \frac{D_n([s, a], d)}{\sum_{a' \in A} D_n([s, a'], d)} = \frac{P^*(d \mid [s, a])P_n([s, a])}{\sum_{a' \in A} P^*(d \mid [s, a'])P_n([s, a'])} = \frac{P^*(d \mid [s, a])P_n(a \mid s)}{\sum_{a' \in A} P^*(d \mid [s, a'])P_n(a' \mid s)}.
$$

The last equality above uses Lemma 2. \qed

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:

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

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 $\theta_{n+1}$ 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:

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

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 $P_{\pi_{n+1}}(a \mid s, d) = D_n(a \mid s, d)$ for every $(s, d) \in \text{supp}D_n(s, d)$. We will refer to this as optimally performing the maximum likelihood estimation at $n^{th}$ epoch. If the learned policy is given by $P_{n+1}(a \mid s, d) = D_n(a \mid s, d)$, then the next Lemma states the relationship between the marginal distribution $P_{n+1}(a \mid s)$ for the next time epoch and marginal $P_n(a \mid s)$ 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 $P_{n+1}(a \mid s)$ for the $(n + 1)^{th}$ epoch is given by:

$$P_{n+1}(a \mid s) = \sum_d \frac{P^*(d \mid [s, a])P_n(a \mid s)P^*(d \mid s)}{\sum_{a' \in A} P^*(d \mid [s, a'])P_n(a' \mid s)}.$$
Proof. The proof is completed as follows:

\[
\mathbb{P}_{n+1}(a \mid s) = \sum_d \mathbb{P}^*(d \mid s) \mathbb{P}_{n+1}(a \mid s, d) = \sum_d \mathbb{P}^*(d \mid [s, a]) \mathbb{P}_n(a \mid s) \mathbb{P}^*(d \mid s),
\]

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 EPOCADEL. For any \( n \in \mathbb{N} \), the \( n^{th} \) epoch of EPOCADEL 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 \( \mathbb{P}^*(a \mid d) \). We then discuss how this can be used to learn the optimal policy \( \mathbb{P}^*(a \mid s, d) \).

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 \( \mathbb{P}_n(d \mid s) := \sum_{a \in A} \mathbb{P}^*(d \mid [s, a]) \mathbb{P}_n(a \mid s) \). Then for every start state \( s \) we have:

\[
D_{KL}(\mathbb{P}^*(a \mid s) \mid \mathbb{P}_{n+1}(a \mid s)) - D_{KL}(\mathbb{P}^*(a \mid s) \mid \mathbb{P}_n(a \mid s)) \leq -D_{KL}(\mathbb{P}^*(d \mid s) \mid \mathbb{P}_n(d \mid s)).
\]

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

\[
D_{KL}(\mathbb{P}^*(a \mid s) \mid \mathbb{P}_{n+1}(a \mid s)) - D_{KL}(\mathbb{P}^*(a \mid s) \mid \mathbb{P}_n(a \mid s)) = -\sum_{a \in A} \mathbb{P}^*(a \mid s) \ln \left( \frac{\mathbb{P}_{n+1}(a \mid s)}{\mathbb{P}_n(a \mid s)} \right)
\]

(12)

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

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

Taking logarithms and applying Jensen’s inequality gives:

\[
\ln \left( \frac{\mathbb{P}_{n+1}(a \mid s)}{\mathbb{P}_n(a \mid s)} \right) = \ln \left( \sum_d \mathbb{P}^*(d \mid [s, a]) \frac{\mathbb{P}^*(d \mid s)}{\mathbb{P}_n(d \mid s)} \right) \geq \sum_d \mathbb{P}^*(d \mid [s, a]) \ln \left( \frac{\mathbb{P}^*(d \mid s)}{\mathbb{P}_n(d \mid s)} \right).
\]

(13)

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

\[
\sum_a \mathbb{P}^*(a \mid s) \ln \left( \frac{\mathbb{P}_{n+1}(a \mid s)}{\mathbb{P}_n(a \mid s)} \right) \geq \sum_a \sum_d \mathbb{P}^*(a \mid s) \mathbb{P}^*(d \mid [s, a]) \ln \left( \frac{\mathbb{P}^*(d \mid s)}{\mathbb{P}_n(d \mid s)} \right)
\]

\[
= \sum_d \mathbb{P}^*(d \mid s) \ln \left( \frac{\mathbb{P}^*(d \mid s)}{\mathbb{P}_n(d \mid s)} \right) = D_{KL}(\mathbb{P}^*(d \mid s) \mid \mathbb{P}_n(d \mid s))
\]

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 \( \mathbb{P}^*(d \mid [s, a]) \). 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 \( \mathbb{P}^*(a \mid s) \) using access to two distributions: \( \mathbb{P}^*(d \mid s) \) which generates the initial requests, and the teacher model \( \mathbb{P}^*(d \mid [s, a]) \) 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 \( \mathbb{P}^*(a \mid s) \) is that \( \sum_{a \in A} \mathbb{P}^*(d \mid [s, a]) \mathbb{P}^*(a \mid s) = \mathbb{P}^*(d \mid s) \). This means all we know is that \( \mathbb{P}^*(a \mid s) \) belongs to the following set of solutions of the previous linear systems of equation:

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

As \( \mathbb{P}^*(a \mid s) \) 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 \( \mathbb{P}_s \mathbb{P}_s^\top \), and so \( \sigma_{\min}(s) > 0 \) implies that the matrix \( \mathbb{P}_s \mathbb{P}_s^\top \) is invertible. 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 \( \mathbb{P}^*(a \mid s) \) 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 \( \mathbb{P}^*(a \mid s) \) at a rate determined by the inverse square root of the number of epochs of \( \text{ADEL} \), 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:

\[
\| \mathbb{P}^*(a \mid s) - \frac{1}{t} \sum_{n=1}^{t} \mathbb{P}_n(a \mid s) \|_2 \leq \frac{1}{\sigma_{\min}(s)} \sqrt{\frac{2}{t} D_{\text{KL}}(\mathbb{P}^*(a \mid s) \| \mathbb{P}_1(a \mid s))},
\]

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

\[
\| \mathbb{P}^*(a \mid s) - \frac{1}{t} \sum_{n=1}^{t} \mathbb{P}_n(a \mid s) \|_2 \leq \frac{1}{\sigma_{\min}(s)} \sqrt{\frac{2 \ln |A|}{t}}.
\]

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

\[
D_{\text{KL}}(\mathbb{P}^*(a \mid s) \| \mathbb{P}_{n+1}(a \mid s)) - D_{\text{KL}}(\mathbb{P}^*(a \mid s) \| \mathbb{P}_n(a \mid s)) \leq -D_{\text{KL}}(\mathbb{P}^*(d \mid s) \| \mathbb{P}_n(d \mid s))
\]

\[
\leq -\frac{1}{2} \| \mathbb{P}^*(d \mid s) - \mathbb{P}_n(d \mid s) \|_2^2,
\]

\[
\leq -\frac{1}{2} \| \mathbb{P}^*(d \mid s) - \mathbb{P}_n(d \mid s) \|_2^2,
\]

\[
= -\frac{1}{2} \| \mathbb{P}_s \{ \mathbb{P}^*(a \mid s) - \mathbb{P}_n(a \mid s) \} \|_2^2,
\]

\[
\leq -\frac{1}{2} \sigma_{\min}(s)^2 \| \mathbb{P}^*(a \mid s) - \mathbb{P}_n(a \mid s) \|_2^2,
\]

where the second step uses Pinsker’s inequality. The third step uses the property of \( p \)-norms, specifically, \( \| \nu \|_2 \leq \| \nu \|_1 \) for all \( \nu \). The fourth step, uses the definition of \( \mathbb{P}^*(d \mid s) = \sum_{a' \in A} \mathbb{P}(d \mid s, a') \mathbb{P}^*(a' \mid s) \) and \( \mathbb{P}_n(d \mid s) = \sum_{a' \in A} \mathbb{P}(d \mid s, a') \mathbb{P}_n(a' \mid s) \). We interpret the notation \( \mathbb{P}^*(a \mid s) \) as a vector over actions whose value is the probability \( \mathbb{P}^*(a \mid s) \). Therefore, \( \mathbb{P}_s \mathbb{P}^*(a \mid s) \) represents a matrix-vector multiplication. Finally, the last step, uses \( \| Ax \|_2 \geq \sigma_{\min}(A) \| x \|_2 \) 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:

\[
D_{\text{KL}}(\mathbb{P}^*(a \mid s) \| \mathbb{P}_{t+1}(a \mid s)) \leq D_{\text{KL}}(\mathbb{P}^*(a \mid s) \| \mathbb{P}_1(a \mid s)) - \frac{1}{2} \sigma_{\min}(s)^2 \sum_{n=1}^{t} \| \mathbb{P}^*(a \mid s) - \mathbb{P}_n(a \mid s) \|_2^2.
\]

\[\text{Recall that a matrix of the form } A^\top A \text{ always have non-negative eigenvalues.}\]
As the left hand-side is positive we get:

\[
\sum_{n=1}^{t} \left\| P^*(a \mid s) - P_n(a \mid s) \right\|^2_2 \leq \frac{2}{\sigma_{\min}(s)^2} D_{KL}(P^*(a \mid s) \mid\mid P_1(a \mid s)).
\]

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

\[
\frac{1}{t} \sum_{n=1}^{t} \left\| P^*(a \mid s) - P_n(a \mid s) \right\|_2 \leq \frac{1}{\sigma_{\min}(s)} \sqrt{\frac{2}{t} D_{KL}(P^*(a \mid s) \mid\mid P_1(a \mid s))}
\]  

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

\[
\frac{1}{t} \sum_{n=1}^{t} \left\| P^*(a \mid s) - P_n(a \mid s) \right\|_2 \geq \left\| P^*(a \mid s) - \frac{1}{t} \sum_{n=1}^{t} P_n(a \mid s) \right\|_2
\]  

Combining the previous two equations proves the main result. Finally, note that if \( P_1(a \mid s, d) = \frac{1}{|A|} \) for every value of \( s, d \), and \( a \), then \( P_1(a \mid s) \) is also a uniform distribution over actions. The initial KL-divergence is then bounded by \( \ln |A| \) as shown below:

\[
D_{KL}(P^*(a \mid s) \mid\mid P_1(a \mid s)) = -\sum_{a \in A} P^*(a \mid s) \ln \frac{1}{|A|} + \sum_{a \in A} P^*(a \mid s) \ln P^*(a \mid s) \leq \ln |A|,
\]

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 \( P^*(a \mid s, d) \) once we learn \( P^*(a \mid s) \). Since we only derive convergence of running average of \( P_n(a \mid s, d) \) to \( P^*(a \mid s) \), therefore, we cannot expect \( P_n(a \mid s, d) \) to converge to \( P^*(a \mid s, d) \). 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 \( P_n(a \mid s, d) \), and once when taking actions according to running average policy \( \tilde{P}_n(a \mid s, d) = \frac{1}{n} \sum_{t=1}^{n} P_n(a \mid s, d) \). This will give us two Bayes optimal policy in 10 one each for the current policy \( P_n(a \mid s, d) \) and the running average policy \( \tilde{P}_n(a \mid s, d) \). 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.

\[ W \text{ operator.} \] Let \( P(a \mid s, d) \) 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 \( (WP) \). Under the realizability assumption (Assumption 1), the agent learns the \( WP \) policy when \( M \to \infty \). Using Lemma 2 and Lemma 3, we can verify that:

\[
(WP)(a \mid s, d) = \frac{P^*(d \mid [s, a])P(a \mid s)}{\sum_{a' \in A} P^*(d \mid [s, a'])P(a' \mid s)}, \quad \text{where} \quad P(a \mid s) = \sum_d P^*(d \mid s)P(a \mid s, d).
\]

We first show that our operator is smooth around \( P^*(a \mid s) \).

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

\[
\|WP(a \mid s, d) - WP^*(a \mid s, d)\|_1 \leq K_s\|P(a \mid s) - P^*(a \mid s)\|_1.
\]
We define $P(d \mid s) = \sum_{a' \in A} P^*(d \mid s, a') P(a' \mid s)$. Then from the definition of operator $W$ we have:

$$|WP(a \mid s, d) - WP^*(a \mid s, d)|_1 = \sum_{a \in A} \left| \frac{P^*(d \mid [s, a]) P(a \mid s) - P^*(d \mid s) P^*(a \mid s)}{P(d \mid s)} \right| \leq \sum_{a \in A} \frac{P^*(d \mid [s, a]) |P(a \mid s) - P^*(a \mid s)|}{P^*(d \mid s)} + \sum_{a \in A} \frac{P^*(d \mid s) |P(a \mid s) - P^*(a \mid s)|}{P^*(d \mid s)} \leq 2 \sum_{a \in A} \frac{P^*(d \mid [s, a]) |P(a \mid s) - P^*(a \mid s)|}{P^*(d \mid s)},$$

(using the definition of $P(d \mid s)$)

Note that the policy will only be called on a given pair of $(s, d)$ if and only if $P^*(d \mid s) > 0$, hence, the constant is bounded.

We define $K_s = \max_d \frac{2}{P^*(d \mid s)}$ where maximum is taken over all descriptions $d \in \text{supp} P^*(d \mid s)$.

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

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

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

From Lemma 8 we have:

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

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

$$(WP^*)(a \mid s, d) = \frac{P^*(d \mid s, a) P^*(a \mid s)}{\sum_{a' \in A} P^*(d \mid s, a') P^*(a' \mid s)} = \frac{P^*(d, a \mid s)}{\sum_{a' \in A} P^*(d, a' \mid s)} = \frac{P^*(d, a \mid s)}{P^*(d \mid s)} = P^*(a \mid s, d).$$

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
Interactive Learning from Activity Description

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. \(|D_{\text{train}}^\text{★}| = 3\)). 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 \textit{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 \(P_{\pi_\omega}(e | s_1)\). 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 \texttt{re.sub(pattern, replace, string)} method as the regular expression compiler. The method replaces every substring of \texttt{string} that matches a regular expression \texttt{pattern} with the string \texttt{replace}. A regular expression predicted by our agent \(\hat{a}_{1:H}\) has the form \"pattern\@replace\", where \texttt{pattern} and \texttt{replace} are strings and \texttt{@} is the at-sign character. For example, given the word \texttt{embolden} and the request \texttt{“replace all n with c”}, the agent should ideally generate the regular expression \texttt{”() (n) ()@c”}. We then split the regular expression by the character \texttt{@} into a string \texttt{pattern} = \texttt{”() (n) ()”} and a string \texttt{replace} = \texttt{”c”}. We execute the Python’s command \texttt{re.sub(‘() (n) ()’, ‘c’, ‘embolden’)} to obtain the output word \texttt{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 \texttt{add an AFTER to the start of words beginning with BEFORE}, where \texttt{AFTER} and \texttt{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
An alternative (naive) implementation of sampling from the mixture
Algorithm 5

A

Approximate marginal
π
implementation has two advantages:

λ
policy
π

regular expressions. These regular expressions are generated using the code provided by Andreas et al. (2018).

∑
executions
{
We extend the performance metric
Simulated Teacher.
simulation, validation, and test sets are disjoint.

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

Compute gradients
10:

11:
Compute losses:
8:
Agent samples
7:
Word samples
5:
Input
4:
Initialize policy
3:
Initialize policy
2:
Initialize policy
1:
Sampling from the mixture is simpler: instead of choosing between
π
and
π
, we always use
π
in Alg 5, which learns a policy
π
such that
π
(e | s1, d) approximates the mixture
P
(e | s1, d) in Alg 3. In each episode, we sample an execution
e
using the policy
π
. Then, similar to Alg 3, we ask the teacher
P
T
for a description of
e
and the use the pair
(e,
(d
) to update the agent policy
π
. To ensure that
π
approximates
P
, we draw a sample
e
from the approximate marginal
P
(e | s1) and update
π
using a
λ
-weighted loss of the log-likelihoods of the two data points
(e,
(d
) and
(e,
(d
). We only use
(e,
(d
) to update the agent policy
π
.

An alternative (naive) implementation of sampling from the mixture
P
is to first choose a policy between
π
(with probability
λ
) and
π
(with probability
1 −
λ
), 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
and
π
, we always use
π
to generate executions;
2. More importantly, samples are more diverse: in the naive approach, the samples are either completely request-agnostic

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
i
i
1
, w
j
o
i
1
}, the metric counts how many pairs where the predicted output word matches the ground-truth:

\[ \sum_{j=1}^{J} \mathbb{1}\{w_{j}^{\text{out}} = w_{j}^{\text{out}}\} \]. We set the threshold
\( \tau = J \).

Approximate marginal
P
(e | s1). 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
such that
\( \mathbb{P}_{\pi_{\alpha}}(e | s_1, d) \) approximates the mixture
\( \hat{P}(e | s_1, d) \) in Alg 3. In each episode, we sample an execution
e
using the policy
π
. Then, similar to Alg 3, we ask the teacher
P
T
for a description of
e
and the use the pair
(e,
(d
) to update the agent policy
π
. To ensure that
P
approximates
\hat{P}
, we draw a sample
e
from the approximate marginal
\( \mathbb{P}_{\pi_{\alpha}}(e | s_1) \) and update
π
using a
λ
-weighted loss of the log-likelihoods of the two data points
(e,
(d
) and
(e,
(d
). We only use
(e,
(d
) to update the agent policy
π
.

\(^{10}\)https://github.com/jacobandreas/l3/blob/master/data/re2/generate.py
Interactive Learning from Activity Description

<table>
<thead>
<tr>
<th>Anneal ( \lambda ) every ( L ) steps</th>
<th>NAV</th>
<th></th>
<th>REGEX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Success rate (%) ↑</td>
<td>Sample complexity ↓</td>
<td>Success rate (%) ↑</td>
<td>Sample complexity ↓</td>
</tr>
<tr>
<td>L = 2000</td>
<td>31.4</td>
<td>304K</td>
<td>87.7</td>
</tr>
<tr>
<td>L = 5000</td>
<td>32.5</td>
<td>384K</td>
<td>86.4</td>
</tr>
<tr>
<td>No annealing (final)</td>
<td>32.0</td>
<td>384K</td>
<td>88.0</td>
</tr>
</tbody>
</table>

Table 5. Effects of annealing the mixing weight \( \lambda \). When annealed, the mixing weight is updated as \( \lambda \leftarrow \max(\lambda_{\min}, \lambda \cdot \beta) \), where the annealing rate \( \beta = 0.5 \) and the minimum mixing rate \( \lambda_{\min} = 0.1 \). Initially, \( \lambda \) 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).

Effects of the Annealing Mixing Weight. We do not anneal the mixing weight \( \lambda \) 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

\[
\frac{|u^\text{out}| - \text{editdistance}(\hat{u}^\text{out}, u^\text{out})}{|u^\text{out}|}
\]

(16)

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

In NAV, the continuous reward function is

\[
\frac{\text{shortest}(s_1, s_g) - \text{shortest}(s_{\text{HI}}, s_g)}{\text{shortest}(s_1, s_g)}
\]

(17)

where \( s_1 \) is the start location, \( s_g \) is the goal location, \( s_{\text{HI}} \) is the agent’s final location, and \( \text{shortest}(\cdot, \cdot) \) 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 \( \tilde{P}(d \mid e) \). 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.
Interactive Learning from Activity Description

Figure 4. Student and teacher models in NAV.

(a) Student model

(b) Teacher model

Figure 5. Student and teacher models in REGEX.
Interactive Learning from Activity Description

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>NAV</th>
<th>REGEX</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Student policy</strong> $\pi_\theta$ and <strong>Teacher’s describer model</strong> $\tilde{P}_T$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Base architecture</td>
<td>Transformer</td>
<td>LSTM</td>
</tr>
<tr>
<td>Hidden size</td>
<td>256</td>
<td>512</td>
</tr>
<tr>
<td>Number of hidden layers (of each encoder or decoder)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Request word embedding size</td>
<td>256</td>
<td>128</td>
</tr>
<tr>
<td>Character embedding size (for the input and output words)</td>
<td>-</td>
<td>32</td>
</tr>
<tr>
<td>Time embedding size</td>
<td>256</td>
<td>-</td>
</tr>
<tr>
<td>Attention heads</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>Observation feature size</td>
<td>2048</td>
<td>-</td>
</tr>
</tbody>
</table>

| **Teacher simulation** |       |       |
| perf metric | STDW (Magalhaes et al., 2019) | Number of output words matching ground-truths |
| Number of samples for approximate pragmatic inference ($|D_{cand}|$) | 5 | 10 |
| Threshold ($\tau$) | 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 ($\lambda$, no annealing) | 0.5 | 0.5 |

Table 6. Hyperparameters for training with the ADEL algorithm.

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

Table 7. Qualitative examples in the REGEX problem. We show pairs of input and output words and how the teacher’s language model $\tilde{P}(d \mid c)$ describes the modifications applied to the input words.
Figure 6. Qualitative examples in the NAV problem. The black texts (no underlines) are the initial requests $d^*$ generated by humans. The $\hat{r}$ paths are the ground-truth paths implied by the requests. $\hat{r}$ and $\hat{r}'$ 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 $\hat{P}(d|e)$. 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.