Bilevel Optimization of Synthetic Trajectories for Multi-Turn LLM Fine-Tuning

Shresth Verma
Harvard University
sverma@g.harvard.edu

Mauricio Tec*
Amazon Science
mauriciogtec@gmail.com

Cheol Woo Kim
Harvard University
cwkim@g.harvard.edu

Kai Wang
Georgia Institute of Technology
kwang692@gatech.edu

Milind Tambe
Harvard University
tambe@g.harvard.edu

Abstract

While LLMs excel at single-turn generation, they struggle with long-horizon, multi-turn interactions. Offline reinforcement learning (RL) offers a scalable approach, yet its performance hinges on the availability and quality of multi-turn trajectory data. A common remedy is to augment training with synthetic trajectories generated by LLMs or simulators, but synthetic data is highly heterogeneous in quality, and naively treating all trajectories as equally informative can degrade performance. We propose BOOST, a bilevel optimization framework where the inner level trains the LLM on reweighted data and the outer level trains a lightweight reweighting head on held-out real validation tasks, assigning continuous trajectory-level weights without requiring an external judge. To ground this approach, we derive a PAC-Bayesian bound revealing a three-way trade-off: synthetic data increases diversity but risks task-shift, while concentrating weight on high-quality trajectories improves empirical performance at the cost of effective sample size. Empirically, our method consistently outperforms multiple baselines. Analysis reveals it upweights synthetic trajectories that align with the real data distribution and exhibit higher qualitative merit.

1 Introduction

Large Language Models (LLMs) excel at generating fluent, human-like text, yet they often struggle with long-horizon reasoning and interaction [Liu et al., 2024, Li et al., 2025, Han et al., 2025, Laban et al., 2026]. In multi-turn settings such as negotiation, persuasion, or technical troubleshooting, effective behavior requires planning across multiple steps, where intermediate actions may not yield immediate rewards but are essential for long-term success. For instance, a capable agent might ask a clarifying question rather than providing a premature answer, prioritizing future outcomes over short-term quality [Aliannejadi et al., 2019, Kuhn et al., 2022, Abdulhai et al., 2025].

To address this limitation, multi-turn interaction can be framed as a reinforcement learning (RL) problem. While online RL optimizes for long-term objectives, model scale and interaction costs make it prohibitively expensive for LLMs. Consequently, offline RL has emerged as a scalable alternative, learning policies from pre-collected trajectories [Snell et al., 2022, Chebotar et al., 2023, Hong et al., 2025]. Despite its scalability advantages, offline RL introduces a new bottleneck: the availability and quality of multi-turn trajectory data. High-performing offline RL methods require datasets with sufficient coverage of the state–action space, which is especially challenging in multi-turn domains.

*Work done while a Postdoctoral Fellow at Harvard University.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

To mitigate this issue, recent work has explored augmenting training data with synthetic trajectories generated by LLMs or simulators [Singh et al., 2023, Goldie et al., 2025]. While this improves coverage, it raises a critical challenge: synthetic data is highly heterogeneous in quality. A naive approach is to treat all trajectories as equally informative, but this can lead to suboptimal or even degraded performance, as low-quality or misleading trajectories may dominate the learning signal [Alemohammad et al., 2023, Shumailov et al., 2023, Gerstgrasser et al., 2024]. Prior work has attempted to address this via binary filtering using a stronger LLM judge, retaining only trajectories that pass a quality threshold, or using expert knowledge base for generating synthetic data [Zelikman et al., 2022, Gunasekar et al., 2023, Goldie et al., 2025]. While effective, such approaches discard potentially useful signal and rely on access to a higher-capacity scorer.

We propose a principled bilevel optimization framework for incorporating synthetic data into multi-turn RL for LLM fine-tuning. Alongside fine-tuning the LLM, we learn a lightweight reweighting head that assigns a scalar weight to each trajectory in a mixed dataset. These normalized weights modulate each trajectory's contribution in standard offline RL objectives, ensuring compatibility with various algorithms. Unlike heuristic filtering, our method performs continuous reweighting based on validation performance. We formulate the learning of reweighting model as a bilevel optimization problem: the inner level trains the policy on reweighted data, while the outer level optimizes the weights to improve performance on held-out real validation tasks. This formulation is similar to decision-focused learning [Donti et al., 2017, Elmachtoub and Grigas, 2022, Wang et al., 2023], which trains predictive models end-to-end for the quality of the downstream decisions they induce. Analogously, BOOST does not treat synthetic data selection as an isolated preprocessing step. Instead, each trajectory's weight is learned according to its effect on the downstream multi-turn decisions of the LLM fine-tuned on it. Further, prior work on data reweighting focuses on coarse-grained data source reweighting [Pan et al., 2025, Xie et al., 2023]. Our method operates at the trajectory level, enabling fine-grained control over data quality. This is particularly important in settings with synthetic data, where variability in quality can be substantial even within a single data source. Empirically, we demonstrate that our approach improves multi-turn RL performance by effectively leveraging synthetic data while mitigating the impact of low-quality trajectories. Analysis shows our model upweights trajectories that align more closely with the real data distribution.

We summarize our contribution as follows:

1. We propose a bi-level optimization framework for trajectory-level reweighting of synthetic data in multi-turn RL fine-tuning of LLMs. We instantiate this framework with a reweighting head trained jointly with the base LLM and the Q functions via a fully first-order optimization scheme. The trajectory-level reweighting head scales to large datasets without prohibitive computational overhead. We further introduce some engineering innovations in bilevel updates which improve training stability.

2. To ground our approach, we derive a PAC-Bayesian bound on performance under trajectory reweighting. This reveals a three-way tradeoff: synthetic data increases diversity but risks task-shift, while focusing on high-quality trajectories improves empirical performance but reduces effective sample size, hiking the complexity penalty.

3. We empirically validate our approach on multi-turn RL benchmarks, showing consistent gains over baselines such as uniform data usage and reweighting applied only to real data. While we anticipated reweighting to always bias toward offline data, analysis reveals a surprise: BOOST selectively upweights synthetic trajectories mirroring the true distribution while suppressing low-quality offline samples. This manages the tradeoff between diversity and task-shift without sacrificing effective sample size. Furthermore, this process sometimes also recovers latent signals from offline points that were undervalued prior to synthetic augmentation, suggesting a synergistic rather than competitive relationship between data sources.

Related Work Related work spans three areas: synthetic data augmentation and its degradation risks [Villalobos et al., 2022, Alemohammad et al., 2023, Shumailov et al., 2023]; bi-level optimization for SFT data reweighting [Pan et al., 2025, Kwon et al., 2023, Chen et al., 2025]; and offline/multi-turn RL for long-horizon LLM planning [Snell et al., 2022, Chebotar et al., 2023, Hong et al., 2025]. BOOST uniquely intersects these by extending bi-level reweighting to multi-turn RL with synthetic augmentation.

2 Preliminaries: Multi-Turn LLMs Fine-Tuning and Data Selection

2.1 Task-conditioned Markov Decision Processes

We model multi-turn LLM dialogue as a *task-conditioned* Markov Decision Process $\mathcal{M}_z = (\mathcal{S}, \mathcal{A}, P_z, R_z, \rho_z, \gamma)$, where each task $z \sim \mathcal{P}_{\text{task}}$ is drawn from a task distribution and indexes a task-specific transition kernel P_z, reward R_z, and initial-state distribution ρ_z [Hallak et al., 2015, Finn et al., 2017]. The state s_t is the full interaction history up to time t; an action a_t is a complete utterance sampled from the policy $\pi(\cdot \mid s_t)$; and the next state concatenates the current state, action, and environment response: $s_{t+1} = [s_t; a_t; o_t]$. In the *infinite-horizon* formulation, the goal is to find a policy π_θ maximizing the expected task-conditioned discounted return $J_\infty(\pi) = \mathbb{E}_{z\sim\mathcal{P}_{\text{task}},\, \tau\sim p_\pi}[\sum_{t=0}^{\infty} \gamma^t R_z(s_t, a_t)]$ with discount factor $\gamma \in (0, 1)$. In the *finite-horizon* formulation, episodes terminate after at most $H_{\max}$ steps. Many RL algorithms are traditionally framed for the infinite horizon case, while being often applied in practice to both frameworks.

The common setup we consider for the theory and experiments in this paper is the task-conditioned finite-horizon setup with horizon $H_{\max}$, reward $R_z(s_t, a_t) = -1$ at every step prior to task completion, $R_z(s_t, a_t) = 0$ once the task is completed (an absorbing terminal state), and $\gamma = 1$; defining the expected hitting time of policy π on task z as $T_z^\pi = \mathbb{E}[\inf\{t \in \mathbb{N} : \text{task } z \text{ completed at step } t\}]$, with $0 \le T_z^\pi \le H_{\max}$, the cumulative return of any trajectory under this specification equals $-T_z^\pi$, so maximizing the expected return $\mathbb{E}_{z\sim\mathcal{P}_{\text{task}}}[J_H(\pi)]$ is equivalent to minimizing the expected hitting time $\mathbb{E}_{z\sim\mathcal{P}_{\text{task}}}[T_z^\pi]$. For notational simplicity, we omit the explicit task index z for the remainder of this section when discussing the offline RL objectives and value functions.

2.2 Offline RL Fine-Tuning for LLMs

In the offline setting, we do not interact with the environment but instead learn from a static dataset $D = \{(s_t, a_t, s_{t+1}, r_t)\}$ collected by a behavior policy π_B. To recover a policy that maximizes long-horizon return without online interaction, we estimate Q-values and V-values from D, which allow the policy to be extracted greedily at inference time. We consider two popular approaches:

Monte Carlo Return (MC) [Abdulhai et al., 2025] regresses a Q-head directly onto discounted rewards-to-go $G_t = \sum_{k\ge t} \gamma^{k-t} r_k$:

$$\mathcal{L}_{\text{MC}} = \mathbb{E}_{(s_t,a_t,G_t)\sim D}\left[\big(Q_\theta(s_t, a_t) - G_t\big)^2\right].$$

Implicit Language Q-Learning (ILQL) [Snell et al., 2022] jointly learns $Q_\theta(s, a)$ and $V_\theta(s)$ by minimizing Bellman error alongside an expectile loss L_2^τ that enforces V–Q consistency:

$$\mathcal{L}_{\text{ILQL}} = \mathbb{E}_{(s,a,r,s')\sim D}\left[\big(r + \gamma V_\theta(s') - Q_\theta(s, a)\big)^2\right] + \mathbb{E}_{(s,a)\sim D}\left[L_2^\tau\big(Q_\theta(s, a) - V_\theta(s)\big)\right],$$

After learning the parameter θ, the final policy is extracted differently depending on the RL objective: under MC, actions are upweighted according to their Q-values, $\pi_\theta(a \mid s) \propto \exp\big(\beta\, Q_\theta(s, a)\big)$, while under ILQL, the policy is extracted using the advantage $A_\theta = Q_\theta(s, a) - V_\theta(s)$, $\pi_\theta(a \mid s) \propto \exp\big(\beta\, A_\theta(s, a)\big)$, where $\beta > 0$ is a temperature parameter in both cases.

2.3 Data Selection via Bilevel Optimization

Fine-tuning LLMs is often challenging due to the data quality issue. Poor data quality can significantly degrade the fine-tuning performance. Prior work [Pan et al., 2025] proposes using $\boldsymbol{w} \in \Delta^m$ to weight m different training sources in $\mathcal{D}_{\text{train}}$ to optimize the validation loss of a validation set $\mathcal{D}_{\text{val}}$:

$$\min_{\boldsymbol{w}}\ \mathcal{L}_{\text{RL}}\big(\theta^*(\boldsymbol{w}); \mathcal{D}_{\text{val}}\big), \qquad \text{s.t. } \theta^*(\boldsymbol{w}) = \arg\min_\theta \mathcal{L}_{\text{train}}(\theta, \boldsymbol{w}; \mathcal{D}_{\text{train}}) \coloneqq \sum\nolimits_i w_i \mathcal{L}_{\text{RL}}(\theta, \mathcal{D}_{\text{train}}^{(i)}), \tag{1}$$

where the loss function $\mathcal{L}_{\text{RL}}$ can be chosen to be either MC or ILQL fine-tuning loss. Motivated by the prior work in bilevel optimization [Kwon et al., 2023, Chen et al., 2025], Equation 1 can be reformulated as a minimax problem and trained using iterative gradient descent.

$$\min_{\theta,\boldsymbol{w}} \max_{\psi}\ \mathcal{L}_{\text{RL}}\big(\theta^*(\boldsymbol{w}); \mathcal{D}_{\text{val}}\big) + \alpha\big(\mathcal{L}_{\text{train}}(\theta, \boldsymbol{w}; \mathcal{D}_{\text{train}}) - \mathcal{L}_{\text{train}}(\psi, \boldsymbol{w}; \mathcal{D}_{\text{train}})\big), \tag{2}$$

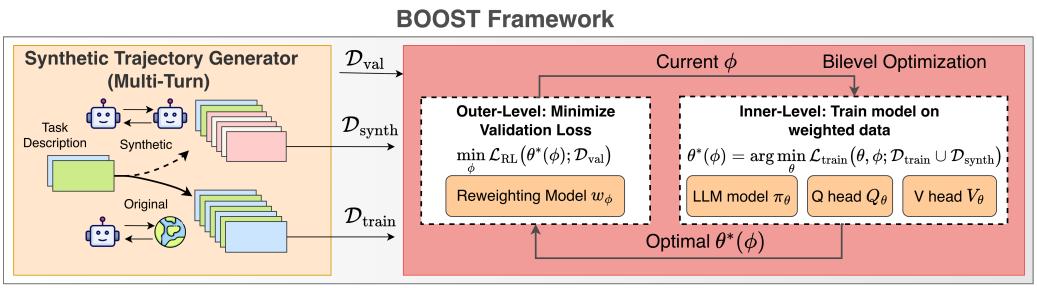

Figure 1: BOOST: The inner level trains model parameters (θ) on a weighted mixture of real and synthetic data, while the outer level updates ϕ to minimize loss on a held-out validation set.

where $\alpha > 0$ is a penalty parameter that enforces the inner-level optimality condition $L_{\text{train}}(\theta, \boldsymbol{w}) \approx L_{\text{train}}(\psi, \boldsymbol{w})$. In practice, researchers have so far limited weight optimization to a few data sources to avoid the instability inherent in LLM fine-tuning. We address this limitation by showing how to transition from these broad sources to scaling directly to granular data samples.

3 BOOST: Bilevel Optimization Of Synthetic Trajectories

To overcome the scarcity of high-quality multi-turn dialogue data, we propose **BOOST** (**B**ilevel **O**ptimization of **S**ynthetic **T**rajectories) (Figure 1). The key idea is to generate synthetic data from the multi-turn conversations in the training set, and learn a parameter-efficient *reweight* model to identify useful synthetic trajectories and give them different training weights to improve downstream RL fine-tuning performance (e.g., under MC, or ILQL losses) on real validation tasks.

3.1 The Challenge of Heterogeneous Synthetic Data

In modern LLM fine-tuning, to ensure sufficient coverage of the vast state-action space, the dataset $\mathcal{D}$ is often augmented such that $\mathcal{D}_{mixed} = \mathcal{D}_{\text{real}} \cup \mathcal{D}_{\text{synthetic}}$. Because synthetic trajectories are generated by simulators or other LLMs, they exhibit extreme heterogeneity in quality. By optimizing the standard offline RL loss over $\mathcal{D}$ uniformly, the policy π_θ is forced to fit low-quality, biased, or ungrounded synthetic trajectories just as strongly as high-quality real data, ultimately degrading the expected return $J(\pi_\theta)$.

3.2 Bilevel Optimization with a Data Reweighting Head

Traditional data mixture methods optimize a sampling vector $\boldsymbol{w} \in \Delta^m$ over m coarse-grained data sources (e.g., different datasets or task categories). In contrast, BOOST learns a parameter-efficient *data reweighting head* w_ϕ that assigns weight to each individual trajectory allowing much finer control over which examples influence learning.

Given $\mathcal{D}_{mixed}$ and a real validation set $\mathcal{D}_{\text{val}}$, we train a language model π_θ using a reinforcement-learning-style loss $\mathcal{L}_{\text{RL}}$ (e.g., MC, or ILQL as defined in Section 2.2). The reweighting model w_ϕ takes as input the a trajectory τ and outputs a weight $w_\phi(\tau)$ that modulates its contribution to the training loss. The weighted training loss is

$$\mathcal{L}_{\text{train}}(\theta, \phi; \mathcal{D}) := \mathbb{E}_{\tau \in \mathcal{D}} \left[\tilde{w}_\phi(\tau) \cdot \mathcal{L}_{\text{RL}}(\theta; \tau) \right],$$

where $\tilde{w}_\phi(\tau)$ denotes normalized weights computed by applying a softmax over the minibatch $\mathcal{B}$: $\tilde{w}_\phi(\tau) = \exp(w_\phi(\tau)) / \sum_{\tau' \in \mathcal{B}} \exp(w_\phi(\tau')))$. BOOST thus learns: (1) The language model parameters θ, by minimizing the reweighted loss on real + synthetic training data. (2) The reweighting parameters ϕ, by encouraging the resulting model $\pi_{\theta^*(\phi)}$ to perform well on real validation data.

3.3 Efficient and Stable Parameter Updates

Algorithm 1 presents the full BOOST training procedure. We introduce three key design choices to ensure stable updates and reliable convergence.

Algorithm 1 BOOST: Data-Reweighted Bilevel Optimization

Require: Training dataset $\mathcal{D}_{\text{train}}$, validation dataset $\mathcal{D}_{\text{val}}$, synthetic data $\mathcal{D}_{\text{synth}}$
Require: Outer iterations N, inner steps K_ψ, K_θ, K_ϕ, Learning rates $\eta_\psi, \eta_\theta, \eta_\phi$; penalty α
Initialize primary policy parameters θ, auxiliary policy parameters ψ, reweighting parameters ϕ
Construct mixed training set $\mathcal{D}_{\text{mixed}} \leftarrow \mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{synth}}$
for $n = 1$ **to** N **do**
  % Phase 1: Update auxiliary policy ψ on mixed training data
  for $j = 1$ **to** K_ψ **do**
    $\psi \leftarrow \psi - \eta_\psi \nabla_\psi \mathcal{L}_{\text{train}}(\psi, \phi; \mathcal{D}_{\text{mixed}})$
  end for
  % Phase 2: Synchronize primary policy with auxiliary copy
  $\theta \leftarrow \psi$
  % Phase 3: Update primary policy θ on validation + mixed training data
  for $j = 1$ **to** K_θ **do**
    $\mathcal{L}_\theta \leftarrow L_{\text{val}}(\theta; \mathcal{D}_{\text{val}}) \; + \; \alpha \, \mathcal{L}_{\text{train}}(\theta, \phi; \mathcal{D}_{\text{mixed}})$
    $\theta \leftarrow \theta - \eta_\theta \nabla_\theta \mathcal{L}_\theta$
  end for
  % Phase 4: Update reweighting model ϕ via joint minimax loss
  for $j = 1$ **to** K_ϕ **do**
    $\mathcal{L}_\phi \leftarrow \mathcal{L}_{\text{val}}(\theta; \mathcal{D}_{\text{val}}) \; + \; \alpha\big(\mathcal{L}_{\text{train}}(\theta, \phi; \mathcal{D}_{\text{mixed}}) - \mathcal{L}_{\text{train}}(\psi, \phi; \mathcal{D}_{\text{mixed}})\big)$
    $\phi \leftarrow \phi - \eta_\phi \nabla_\phi \mathcal{L}_\phi$
  end for
end for

(1) Alternating Frozen Updates. At each inner iteration, we update only one parameter group at a time, the auxiliary policy ψ, the primary policy θ, or the reweighting head ϕ, while keeping the others frozen. This decoupling prevents conflicting gradient signals from destabilizing training. Crucially, ϕ never receives gradients from a simultaneously shifting policy, and policy updates always see a fixed data weighting.

(2) Warm-Start Initialization of θ from ψ. Unlike Kwon et al. [2023], Pan et al. [2025], where θ and ψ are updated independently, we re-initialize $\theta \leftarrow \psi$ at the start of each outer iteration. Since ψ tracks the inner-level optimum of $\mathcal{L}_{\text{train}}(\cdot, \phi)$, this gives θ a warm start that is already near the inner optimum, accelerating convergence and improving stability. This modification introduces no additional computational cost and requires no change to any gradient update rules.

(3) Asymmetric Learning Rates Across Levels. The policy parameters (θ, ψ) use a significantly higher learning rate than the reweighting head ϕ, with ϕ updated up to an order of magnitude more slowly. This reflects the hierarchical structure of bilevel optimization: the lower-level policy must reach a stable, near-optimal solution before the upper-level weights are adjusted. Updating ϕ too quickly before the policy has settled introduces spurious gradients that drive the reweighter toward degenerate solutions.

4 Theoretical Analysis: Trajectory Reweighting and Task Generalization

As established in Section 2, we adopt the finite-horizon, task-conditioned hitting-time specification: tasks z are drawn from $\mathcal{P}_{\text{task}}$, each inducing a task-specific MDP $\mathcal{M}_z$, and T_z^π denotes the expected hitting time of policy π on task z, with $0 \le T_z^\pi \le H_{\max}$. Recall that, under this specification, maximizing expected return is equivalent to minimizing $\mathbb{E}_{z \sim \mathcal{P}_{\text{task}}}[T_z^\pi]$, so the analysis below bounds the latter.

PAC–Bayesian bounds compare a learned predictor to a fixed prior chosen before training. In our setting, the natural prior is the current pretrained LLM before task-specific RL fine-tuning. We denote by P_0 the corresponding prior over policies, centered at the pretrained LLM π_0.

Let $\mathcal{Q}_\theta$ denote the posterior over stochastic policies, induced by the fine-tuned LLM with weights θ. To keep notation compact, write $T_z^{\pi_\theta} := \mathbb{E}_{\pi \sim \mathcal{Q}_\theta}[T_z^\pi]$. Let $\mathcal{P}_\phi$ denote the weighted mixed task

distribution induced by the fixed trajectory weights $\tilde{w}_\phi$ over real and synthetic data. Define

$$\Delta_{\text{task}}(\phi) = \sup_{\pi \in \Pi} \left| \mathbb{E}_{z \sim \mathcal{P}_{\text{task}}}[T_z^\pi] - \mathbb{E}_{z \sim \mathcal{P}_\phi}[T_z^\pi] \right|.$$

This term measures how much the weighted mixed distribution differs from the target task distribution for the policy class under consideration.

We define effective sample size as $n_{\text{eff}}(\tilde{w}_\phi) = \frac{1}{(\sum_{i=1}^n \tilde{w}_{\phi,i}^2)}$, where $\tilde{w}_{\phi,i}$ are trajectory weights.

Theorem 4.1 (Weight-conditional PAC–Bayesian hitting-time bound)**.** *Fix normalized weights $\tilde{w}_\phi$ over $\mathcal{D}_{\text{mixed}}$. Let $\mathcal{P}_0$ be the pretrained-policy prior (chosen independently of the finetuning data), and let $\mathcal{Q}_\theta$ be the posterior induced by the learned policy. Then there exists a universal constant $C_1 > 0$ such that, for any $\delta \in (0,1)$, with probability at least $1-\delta$,*

$$\mathbb{E}_{z \sim \mathcal{P}_{\text{task}}}[T_z^{\pi_\theta}] \leq \underbrace{\sum_{i=1}^n \tilde{w}_{\phi,i} T_{z_i}^{\pi_\theta}}_{\textit{weighted empirical time}} + \underbrace{C_1 H_{\max} \sqrt{\frac{\mathrm{KL}(\mathcal{Q}_\theta \| \mathcal{P}_0) + \log(2/\delta)}{n_{\text{eff}}(\tilde{w}_\phi)}}}_{\textit{policy complexity}} + \underbrace{\Delta_{\text{task}}(\phi)}_{\textit{task-shift penalty}}. \tag{3}$$

Proof sketch. For fixed weights, apply the PAC–Bayesian Hoeffding–Azuma inequality of Seldin et al. [2012] to the bounded loss $T_z^\pi / H_{\max} \in [0,1]$. The weighted centered losses form martingale differences, and the ith increment has range length $\tilde{w}_{\phi,i}$. Thus the Hoeffding–Azuma range term is $\sum_i \tilde{w}_{\phi,i}^2 = 1/n_{\text{eff}}(\tilde{w}_\phi)$. Multiplying by $H_{\max}$ gives a bound on $\mathbb{E}_{z \sim \mathcal{P}_\phi}[T_z^{\pi_\theta}]$ in terms of the weighted empirical hitting time and the PAC–Bayesian complexity term. Finally,

$$\mathbb{E}_{z \sim \mathcal{P}_{\text{task}}}[T_z^{\pi_\theta}] \leq \mathbb{E}_{z \sim \mathcal{P}_\phi}[T_z^{\pi_\theta}] + \Delta_{\text{task}}(\phi).$$

This gives Eq. (3). The proof is in Appendix B, which shows a slightly sharper statement by applying the PAC–Bayesian Hoeffding–Azuma inequality of Seldin et al. [2012]. □

The bound is stated for a fixed weight vector $\tilde{w}_\phi$ and analyzes the inner learning problem once the outer BOOST reweighting model has produced weights. Adding synthetic data uniformly can increase coverage and effective sample size, but only when those trajectories have low hitting time and do not increase the task-shift penalty. Conversely, placing nearly all mass on a small number of apparently good trajectories can reduce the empirical term but lowers n_{eff}, increasing the complexity penalty. BOOST aims for the useful middle regime: it assigns weight to synthetic trajectories that improve validation performance, discounts trajectories that introduce poor behavior or task shift, and preserves enough effective sample size for the resulting policy to generalize to new tasks.

5 Experiments

We evaluate BOOST on three multi-turn dialogue environments: two from the **LMRL-Gym** benchmark [Abdulhai et al., 2025], testing strategic reasoning and sequential decision-making through natural language, plus a benchmark for multi-turn LLM tool use in a realistic web navigation setting. Although each environment specifies its reward in its own natural units (questions-remaining, turns-remaining, or step-discounted match score), in all three cases the reward is equivalent to the finite-horizon hitting-time setup of Section 2.1.

Guess (20 Questions). A strategic information-gathering task from LMRL-Gym with 40K offline dialogues where an agent tries to identify a hidden object by posing yes/no questions to an oracle. Each question receives a binary response; the agent must iteratively narrow the hypothesis space to find the object. Effective policies ask maximally informative questions to efficiently eliminate candidate hypotheses. *Reward:* The agent receives points equal to questions remaining upon correct identification; incorrect guesses or exhausting the limit yield zero reward.

Car Dealer Negotiation. An LMRL-Gym bargaining task with 4.4K offline dialogues where the agent acts as a car salesperson negotiating with a buyer. The agent proposes prices and responds to buyer counteroffers across multiple dialogue turns until an agreement or rejection occurs. The task requires strategic persuasion and principled price management. *Reward:* Rewards depend on the final negotiation outcome and how quickly it is reached: reward equals the number of dialogue turns remaining at the point of agreement; failed negotiations yield zero reward.

WebShop Agent. WebShop [Yao et al., 2022] is an online shopping benchmark featuring diverse structured and unstructured text. This including product titles, descriptions, and attributes—and requires the agent to complete purchase tasks based on natural language queries (e.g., *"I am looking for a nightstand with drawers, a nickel finish, and a price below $140"*). The agent has multi-turn interaction with a simulated webpage through search queries, clicks, filter selections, and page navigation. We adapted the original online benchmark for our offline setting, we use GPT-4 to collect 2,000 realistic interaction trajectories as training data. *Reward:* Performance is measured using two metrics: attribute match score (the proportion of desired product attributes satisfied, averaged across episodes) and success rate (the percentage of episodes in which all requirements are fully met). The reward signal used during training is the attribute match score, discounted by steps to checkout.

Data Generation To generate synthetic data $\mathcal{D}_{synth}$ for multi-turn RL, we use offline trajectories as a starting point. For each environment, we initialize from an offline trajectory's first two steps, then simulate the remainder by prompting an LLM to alternate roles as agent and environment. The trajectory ends at the turn limit or natural conclusion. No external simulator is used. To ensure a fair comparison, we use the same LLM that is subsequently fine-tuned for data generation.

A central goal of this work is to evaluate generalization from limited training data to unseen task categories. Thus, we partition the data by task category rather than by individual task instance. In the 20 Questions task, categories correspond to the semantic class of the target word (e.g., tools, fruits); in Car Dealer Negotiation, they correspond to car brands, which implicitly encode information such as price range and feature set; in WebShop, they correspond to product classes (e.g., beauty products, apparel). Offline trajectories and the synthetic data derived from them are split by category into train, validation, and evaluation sets. The validation set shares the same category distribution as the evaluation set but is kept deliberately small, reflecting realistic conditions in which only limited labeled data from target tasks is available. This setup directly tests the model's ability to generalize across task categories rather than merely interpolate within them. Further details in the Appendix.

5.1 Evaluation

For each environment, we train using two RL algorithms: Monte Carlo Regression and ILQL. Within each algorithm, we evaluate four methods:

1. **Vanilla.** Training uses only the available offline dataset.
2. **Vanilla+Synthetic.** Training uses the offline dataset augmented with synthetic trajectories.
3. **Bilevel Only.** Training uses only offline data, but optimizes sample weights via bilevel updates on the train/validation split.
4. **BOOST.** Training uses offline data augmented with synthetic trajectories, optimized via the bilevel procedure described in Algorithm 1.

To assess the effect of data availability, each regime is evaluated under both a **high-data** and a **low-data** setting, where the low-data setting uses an order of magnitude fewer training examples than the high-data setting. This directly probes each method's robustness under the limited training data conditions that motivate this work.

While all methods train on offline data, policies are evaluated by interacting with a live simulated environment over a set of held-out goals. This constitutes a deliberately challenging evaluation regime that tests generalisation across unseen task categories and exposes each policy to the distribution shift inherent in moving from static, logged trajectories to dynamic online interaction where the policy's own outputs shape subsequent environment responses in ways not observed during training, more closely reflecting the conditions these policies would face in real-world deployment.

For the 20 Questions and Car Dealer Negotiation tasks, we fine-tune a GPT-2 model, which has been shown to perform adequately on these tasks in prior work Abdulhai et al. [2025]. WebShop is a substantially more challenging environment, requiring multi-step information gathering and diverse action types. Thus, we fine-tune a Llama-3-8B model and a Qwen-3.5-9B model using parameter-efficient fine-tuning via QLoRA. To ensure fair comparison across methods, the LLM used to generate synthetic data is always identical to the one being fine-tuned. Further implementation details are in the Appendix.

Table 1: Evaluation reward of MC Return and ILQL across environments, LLM models, and data regimes (mean $\pm$ standard error over 64 evaluation rollouts; higher is better). Bilevel reweighting outperforms vanilla in nearly all settings, and BOOST is the best method in all low-data settings.

Data	Algo	Environment (LLM)	Vanilla	Vanilla+Syn	Bilevel Only	BOOST (ours)
Low	MC	WebShop (Qwen 3.5-9B)	0.23 ± 0.06	0.28 ± 0.04	0.41 ± 0.04	$\mathbf{0.43 \pm 0.03}$
		WebShop (Llama-3-8B)	0.24 ± 0.02	0.23 ± 0.02	0.32 ± 0.03	$\mathbf{0.40 \pm 0.03}$
		Car Negotiation (GPT-2)	26.73 ± 0.39	31.08 ± 0.00	36.28 ± 2.23	$\mathbf{40.92 \pm 1.86}$
		Twenty Questions (GPT-2)	1.01 ± 0.13	1.01 ± 0.11	2.90 ± 0.22	$\mathbf{3.46 \pm 0.22}$
	ILQL	WebShop (Qwen 3.5-9B)	0.30 ± 0.05	0.33 ± 0.04	0.33 ± 0.04	$\mathbf{0.40 \pm 0.04}$
		WebShop (Llama-3-8B)	0.30 ± 0.02	0.27 ± 0.02	0.32 ± 0.03	$\mathbf{0.38 \pm 0.02}$
		Car Negotiation (GPT-2)	27.22 ± 0.19	25.94 ± 0.93	29.08 ± 2.01	$\mathbf{34.91 \pm 2.37}$
		Twenty Questions (GPT-2)	0.38 ± 0.08	0.45 ± 0.09	0.51 ± 0.11	$\mathbf{0.53 \pm 0.10}$
High	MC	WebShop (Llama-3-8B)	0.38 ± 0.02	0.31 ± 0.02	0.43 ± 0.03	$\mathbf{0.45 \pm 0.03}$
		Car Negotiation (GPT-2)	32.27 ± 0.39	35.09 ± 0.39	41.31 ± 1.71	$\mathbf{41.42 \pm 1.71}$
		Twenty Questions (GPT-2)	2.22 ± 0.18	2.19 ± 0.23	$\mathbf{3.83 \pm 0.23}$	3.48 ± 0.21
	ILQL	WebShop (Llama-3-8B)	0.31 ± 0.02	0.27 ± 0.02	$\mathbf{0.41 \pm 0.02}$	0.36 ± 0.02
		Car Negotiation (GPT-2)	35.33 ± 0.78	33.00 ± 0.78	$\mathbf{37.55 \pm 2.40}$	37.36 ± 2.39
		Twenty Questions (GPT-2)	2.60 ± 0.21	2.04 ± 0.19	5.38 ± 0.38	$\mathbf{5.83 \pm 0.28}$

Table 2: Data-selection baselines vs. BOOST in the low-data regime (evaluation reward, higher is better). WebShop uses Llama-3-8B; Car Negotiation and Twenty Questions (20Q) use GPT-2.

Method	Web MC	Web ILQL	Car MC	Car ILQL	20Q MC	20Q ILQL
Vanilla+Syn	0.23	0.27	31.08	25.94	1.01	0.45
Binary filter [Singh et al., 2023]	0.25	0.26	20.08	22.58	2.18	0.50
LLM-agent weighting [Chen et al., 2024]	0.23	0.27	19.94	19.00	1.06	**0.53**
ScaleBiO [Pan et al., 2025]	0.35	0.33	**41.58**	29.06	2.80	0.51
BOOST (ours)	**0.40**	**0.38**	40.92	**34.91**	**3.46**	**0.53**

6 Results and Discussion

In Table 1, we report test-time reward performance for the MC Returns and ILQL fine-tuning methods averaged over 64 evaluation rollouts. We discuss several key findings below.

Bilevel reweighting consistently improves performance. Across both fine-tuning methods, all three environments, and both data regimes, BOOST and Bilevel strictly outperform both vanilla versions. This consistent pattern strongly suggests that offline datasets are heterogeneous in quality; reweighting examples to optimize downstream validation performance is a reliable and general improvement over uniform training. The margins are often substantial: in the MC Returns setting , BOOST improves over both vanilla settings by a substantial margin in the low-data regime, successfully completing the 20 Questions task an average of ~ 2 turns earlier and the Car Dealer Negotiation an average of ~ 9 turns earlier. This improvement is also statistically significant.

Naively adding synthetic data can hurt . While more data generally aids generalization, unweighted synthetic augmentation is unreliable. In the ILQL setting, Vanilla with Synthetic underperforms Vanilla in four out of seven settings, Car Negotiation high-data (33.00 vs 35.33), low-data (25.94 vs 27.22) and both Webshop regimes (0.27 vs 0.30/0.31). A similar pattern appears in MC Returns, Vanilla with Synthetic performs slightly poorly in Twenty Questions high-data (2.19 vs 2.22) and Webshop high-data (0.31 vs 0.38). This demonstrates that synthetic data introduces noise that, without proper reweighting, the model cannot filter out.

Comparison with Data-Selection Baselines We compare BOOST against three data-selection baselines in the low-data regime of all environments, for both MC and ILQL. **ScaleBiO source weighting** [Pan et al., 2025] learns two scalar weights, one for offline and one for synthetic trajectories,



(a) Twenty Questions (b) Car price Negotiation. (c) Webshop Agent

(d) Twenty Questions (e) Car Price Negotiation (f) Webshop Agent

Figure 2: Learned weight distributions across environments. *Top:* relative weight change from uniform initialization for synthetic and offline points. *Bottom:* weight change vs. distance to nearest offline neighbours.

using the same bilevel objective on the same real validation data; it differs from BOOST only in reweighting granularity. **Binary filtering**, following Singh et al. [2023], discards synthetic trajectories judged unsuccessful by the same generator that produced them, then trains on the union of offline data and the retained synthetic data. **LLM-agent weighting**, following Chen et al. [2024], uses a strong external LLM agent (Claude Opus 4.8) to identify and reweight low-quality synthetic trajectories; the resulting weights are used as sampling weights during fine-tuning.

Table 2 shows that BOOST is the best or tied-best method in five of six settings. BOOST significantly outperforms Binary filter in all settings and performs significantly better than ScaleBiO and Agent-based reweighting in the majority of settings. Notably, ScaleBiO uses the same validation-driven bilevel objective on the same data, so BOOST's gain comes specifically from per-trajectory granularity, not merely from the presence of weighting. Both fixed-signal heuristics (binary success-filtering and strong LLM agent quality scoring) perform poorly in most settings. This shows that neither a reward heuristic nor a strong external LLM-agent substitutes for learned reweighting.

BOOST achieves strongest results in low-data regimes. By combining bilevel reweighting with synthetic augmentation, BOOST yields the best performance in 11 out of 14 settings. The advantage is most pronounced in the low-data regime, where it is the clear winner across all eight settings, outperforming even Bilevel thus demonstrating that synthetic data provides genuine additional value when reweighted appropriately. The biggest gains are under MC Returns where BOOST improves over the best vanilla baseline by 242.6% in Twenty Questions (3.46 vs 1.01), 31.7% in Car Negotiation (40.92 vs. 31.08), and 66.7% in Webshop (0.40 vs. 0.24). This confirms our motivation: when real trajectories are scarce, synthetic data increases conversational diversity, while bilevel reweighting ensures that only synthetic transitions genuinely aligned with the validation objective receive high weight.

Analyzing Learnt Weights *Weights are non-uniform and vary on a per-point basis.* Initially, all data points are assigned uniform sampling weights. Over the course of BOOST training, the data reweighting head w_ϕ learns heterogeneous weights across data points. In Figures 2a–2c, we plot the relative change in weights from the uniform initialization, grouped separately for synthetic and offline data. Two observations stand out. First, the learned weights vary substantially across data points, confirming the bilevel objective identifies differential utility rather than collapsing to a uniform solution. Second, weights do not separate by source: some synthetic points receive high weight while some offline points are downweighted. This demonstrates that w_ϕ cannot be approximated by any simple heuristic that uniformly downweights synthetic data and upweights offline data. Instead, the head performs fine-grained, per-point discrimination based on validation signals. This underscores the need for BOOST's learned reweighting over simpler source-based filtering or fixed mixing ratios.

Highly-weighted synthetic points are those closest to the offline distribution. In Figures 2d–2f, we test the hypothesis that synthetic data points receiving high weight are those most similar to true offline data points. For each synthetic data point, we plot the relative weight change from uniform against its mean L_1 distance in embedding space to its $K = 10$ nearest offline neighbours. Across all environments, weight and distance from offline data are inversely related: synthetic points closer to the offline distribution receive higher weights, while outliers are discounted. Pearson correlations confirm this trend, showing negative correlations of -0.26 (Twenty Questions), -0.49 (Car Dealer), and -0.10 (Webshop). This evidence confirms BOOST doesn't indiscriminately upweight synthetic data; instead, it selectively incorporates transitions most consistent with real offline data.

Adding synthetic data rescues informative offline points. Finally, we examine how the weights assigned to offline data points shift when synthetic data is introduced into BOOST. Concretely, we compare offline point weights under Bilevel-Offline-Only against their weights under BOOST, and ask whether points that were highly weighted under the offline-only objective remain highly weighted when synthetic data is added. In the Car Dealer environment (see Figure 3), 88% of highly-weighted offline points retain high weight under BOOST, demonstrating that the bilevel objective is stable and does not discard previously identified high-quality transitions. More strikingly, among offline points that received low weight under the offline-only objective, BOOST promotes 36.2% of them to high weight. This suggests that these points possessed latent signal that was previously obscured, likely due to distributional gaps in the offline dataset, and that the additional coverage provided by synthetic data gives the bilevel optimizer sufficient context to recognise and recover their value.

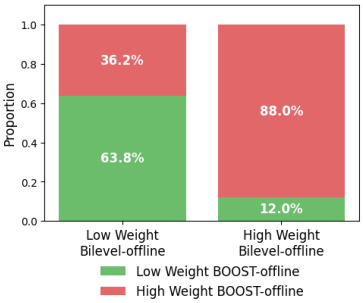

Figure 3: Car Dealer environment weight shift from Bilevel to BOOST

In addition to these three analyses, we provide qualitative comparison of synthetic trajectories with high and low weights in Appendix I . We also show ablations on BOOST's design choices in Appendix G. Together, they showcase that BOOST learns to upweight offline data that is inherently informative, selectively incorporates synthetic data that resembles the offline distribution, and recovers signal from offline points that were undervalued in the absence of synthetic augmentation.

7 Conclusion

We presented BOOST, a bilevel framework for improving multi-turn offline RL fine-tuning by learning trajectory-level weights over mixed real and synthetic data. Across three environments and two RL algorithms, BOOST consistently outperforms uniform training and static synthetic augmentation, with the largest gains in low-data regimes. Analysis shows the learned weights are semantically meaningful: the reweighting head upweights synthetic trajectories nearest to the real distribution, recovers signal from undervalued offline points, and suppresses noisy transitions without explicit quality supervision. Together with our PAC-Bayesian hitting-time bound, these results establish principled synthetic-data reweighting as a practical path toward data-efficient multi-turn LLM adaptation. A current limitation is that reweighting operates at the trajectory level; finer-grained per-timestep reweighting, which could recover useful signal from partially good trajectories, is a natural direction for future work.

Acknowledgments and Disclosure of Funding

This work was supported by ONR MURI N00014-24-1-2742, by NSF IIS-2403240, NSF IIS-2552007, Schmidt Sciences AI2050 Fellowship and based upon work supported by the AI Research Institutes Program funded by the National Science Foundation under the AI Institute for Societal Decision Making (NSF AI-SDM), Award No. 2229881.

References

Marwa Abdulhai, Isadora White, Charlie Victor Snell, Charles Sun, Joey Hong, Yuexiang Zhai, Kelvin Xu, and Sergey Levine. Lmrl gym: Benchmarks for multi-turn reinforcement learning

with language models. In *International Conference on Machine Learning*, pages 126–153. PMLR, 2025.

Sina Alemohammad, Josue Casco-Rodriguez, Lorenzo Luzi, Ahmed Imtiaz Humayun, Hossein Babaei, Daniel LeJeune, Ali Siahkoohi, and Richard Baraniuk. Self-consuming generative models go mad. In *The Twelfth International Conference on Learning Representations*, 2023.

Mohammad Aliannejadi, Hamed Zamani, Fabio Crestani, and W Bruce Croft. Asking clarifying questions in open-domain information-seeking conversations. In *Proceedings of the 42nd international acm sigir conference on research and development in information retrieval*, pages 475–484, 2019.

Olivier Catoni. *PAC-Bayesian Supervised Classification: The Thermodynamics of Statistical Learning*, volume 56 of *Lecture Notes–Monograph Series*. Institute of Mathematical Statistics, 2007.

Yevgen Chebotar, Quan Vuong, Karol Hausman, Fei Xia, Yao Lu, Alex Irpan, Aviral Kumar, Tianhe Yu, Alexander Herzog, Karl Pertsch, et al. Q-transformer: Scalable offline reinforcement learning via autoregressive q-functions. In *Conference on Robot Learning*, pages 3909–3928. PMLR, 2023.

Lesi Chen, Yaohua Ma, and Jingzhao Zhang. Near-optimal nonconvex-strongly-convex bilevel optimization with fully first-order oracles. *Journal of Machine Learning Research*, 26(109):1–56, 2025.

Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, et al. Alpagasus: Training a better alpaca with fewer data. In *International Conference on Learning Representations*, volume 2024, pages 34767–34797, 2024.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115, 2023.

Priya Donti, Brandon Amos, and J Zico Kolter. Task-based end-to-end model learning in stochastic optimization. *Advances in neural information processing systems*, 30, 2017.

Adam N Elmachtoub and Paul Grigas. Smart "predict, then optimize". *Management science*, 68(1): 9–26, 2022.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.

Pascal Germain, Alexandre Lacasse, François Laviolette, and Mario Marchand. PAC-Bayesian learning of linear classifiers. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 353–360, 2009. doi: 10.1145/1553374.1553419.

Matthias Gerstgrasser, Rylan Schaeffer, Apratim Dey, Rafael Rafailov, Henry Sleight, John Hughes, Tomasz Korbak, Rajashree Agrawal, Dhruv Pai, Andrey Gromov, et al. Is model collapse inevitable? breaking the curse of recursion by accumulating real and synthetic data. *arXiv preprint arXiv:2404.01413*, 2024.

Anna Goldie, Azalia Mirhoseini, Hao Zhou, Irene Cai, and Christopher D Manning. Synthetic data generation and multi-step reinforcement learning for reasoning and tool use. In *Second Conference on Language Modeling*, 2025. URL `https://openreview.net/forum?id=oN9STRYQVa`.

Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, et al. Textbooks are all you need. *arXiv preprint arXiv:2306.11644*, 2023.

Assaf Hallak, Dotan Di Castro, and Shie Mannor. Contextual Markov decision processes. *arXiv preprint arXiv:1502.02259*, 2015.

Chi Han, Xin Liu, Haodong Wang, Shiyang Li, Jingfeng Yang, Haoming Jiang, Zhengyang Wang, Qingyu Yin, Liang Qiu, Changlong Yu, et al. Can language models follow multiple turns of entangled instructions? *arXiv preprint arXiv:2503.13222*, 2025.

Joey Hong, Anca Dragan, and Sergey Levine. Planning without search: Refining frontier LLMs with offline goal-conditioned RL. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=kuoD6G0Suq.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Clam: Selective clarification for ambiguous questions with generative language models. *arXiv preprint arXiv:2212.07769*, 2022.

Jeongyeol Kwon, Dohyun Kwon, Stephen Wright, and Robert D Nowak. A fully first-order method for stochastic bilevel optimization. In *International Conference on Machine Learning*, pages 18083–18113. PMLR, 2023.

Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. LLMs get lost in multi-turn conversation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=VKGTGGcwl6.

Yubo Li, Xiaobin Shen, Xinyu Yao, Xueying Ding, Yidi Miao, Ramayya Krishnan, and Rema Padman. Beyond single-turn: A survey on multi-turn interactions with large language models. *arXiv preprint arXiv:2504.04717*, 2025.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the association for computational linguistics*, 12:157–173, 2024.

David A. McAllester. Some PAC-Bayesian theorems. *Machine Learning*, 37(3):355–363, 1999. doi: 10.1023/A:1007618624809.

Rui Pan, Dylan Zhang, Hanning Zhang, Xingyuan Pan, Minrui Xu, Jipeng Zhang, Renjie Pi, Xiaoyu Wang, and Tong Zhang. Scalebio: Scalable bilevel optimization for llm data reweighting. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 31959–31982, 2025.

Matthias Seeger. Pac-bayesian generalisation error bounds for gaussian process classification. *Journal of Machine Learning Research*, 3:233–269, 2002.

Yevgeny Seldin, François Laviolette, Nicolò Cesa-Bianchi, John Shawe-Taylor, and Peter Auer. Pac-bayesian inequalities for martingales. *IEEE Transactions on Information Theory*, 58(12): 7086–7093, 2012. doi: 10.1109/TIT.2012.2216545.

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Yarin Gal, Nicolas Papernot, and Ross Anderson. The curse of recursion: Training on generated data makes models forget. *arXiv preprint arXiv:2305.17493*, 2023.

Avi Singh, John D Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J Liu, James Harrison, Jaehoon Lee, Kelvin Xu, et al. Beyond human data: Scaling self-training for problem-solving with language models. *arXiv preprint arXiv:2312.06585*, 2023.

Charlie Snell, Ilya Kostrikov, Yi Su, Mengjiao Yang, and Sergey Levine. Offline rl for natural language generation with implicit language q learning. *arXiv preprint arXiv:2206.11871*, 2022.

Pablo Villalobos, Jaime Sevilla, Lennart Heim, Tamay Besiroglu, Marius Hobbhahn, and Anson Ho. Will we run out of data? an analysis of the limits of scaling datasets in machine learning. *arXiv preprint arXiv:2211.04325*, 1(1), 2022.

Kai Wang, Shresth Verma, Aditya Mate, Sanket Shah, Aparna Taneja, Neha Madhiwalla, Aparna Hegde, and Milind Tambe. Scalable decision-focused learning in restless multi-armed bandits with application to maternal and child health. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 12138–12146, 2023.

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy Liang, Quoc V. Le, Tengyu Ma, and Adams Wei Yu. Doremi: optimizing data mixtures speeds up language model pretraining. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, 2022.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

A Background on PAC–Bayesian Bounds

PAC–Bayesian analysis gives high-probability generalization bounds for data-dependent randomized predictors [McAllester, 1999, Seeger, 2002, Catoni, 2007, Germain et al., 2009]. Let Π be a policy class, let P_0 be a prior distribution over Π, and let Q be a posterior distribution over Π chosen after observing the training sample. The prior P_0 must be fixed independently of the sample used in the bound. For a bounded loss $\ell(z, \pi) \in [0, 1]$, define

$$L(Q) = \mathbb{E}_{\pi\sim Q}\mathbb{E}_{Z\sim\mathcal{P}}[\ell(Z, \pi)], \qquad \widehat{L}(Q) = \mathbb{E}_{\pi\sim Q}\left[\frac{1}{n}\sum_{i=1}^{n}\ell(Z_i, \pi)\right].$$

In the standard i.i.d. setting, where $Z_1, \ldots, Z_n \sim \mathcal{P}$, PAC–Bayesian bounds control $L(Q)$ by $\widehat{L}(Q)$ plus a complexity term of the form

$$O\left(\sqrt{\frac{\mathrm{KL}(Q\|P_0) + \log(1/\delta)}{n}}\right).$$

The KL term measures how far the learned posterior moves from the prior, while the $1/\sqrt{n}$ factor comes from concentration of the uniform empirical average.

In our setting, predictors are stochastic policies. The natural prior is a distribution over policies centered at the pretrained LLM before task-specific RL fine-tuning. This is the reference policy available before the task-specific trajectories are observed. The posterior Q_θ is the distribution over policies induced by the fine-tuned LLM. Thus $\mathrm{KL}(Q_\theta\|P_0)$ measures deviation from pretrained behavior, rather than deviation from a uniform distribution over utterances.

The technical issue is that BOOST does not use the uniform empirical loss. For fixed normalized weights $w \in \Delta_n$, the empirical loss is

$$\widehat{L}_w(Q) = \mathbb{E}_{\pi\sim Q}\left[\sum_{i=1}^{n} w_i\ell(Z_i, \pi)\right].$$

The standard i.i.d. PAC–Bayesian bound does not directly apply to this weighted average by simply replacing n with an effective sample size. The concentration term must account for the nonuniform ranges of the weighted summands.

Section B handles this point formally. We apply the PAC–Bayesian Hoeffding–Azuma inequality of Seldin et al. [2012] to the weighted centered losses. In the independent-sample setting considered here, these centered losses form martingale differences, and their range lengths are exactly the weights w_i. This yields the weighted PAC–Bayesian bound in Corollary B.2, the exact Seldin-form hitting-time bound in Theorem B.3, and the simplified constant-form bound used in the main text in Corollary B.4.

Remark A.1 (On the independence assumption)**.** *Corollary B.2 assumes $Z_i \perp (Z_1, \ldots, Z_{i-1})$. In our experimental protocol (Section 4 data generation), synthetic trajectories are initialized from prefixes of real trajectories, so some synthetic tasks z_j^{synth} are not independent of real tasks z_i^{real}. The martingale construction extends to this case by placing all real tasks first in the filtration $\{\mathcal{F}_i\}$ and treating synthetic tasks as conditionally independent given the real-task σ-algebra $\mathcal{G}$; $\mathcal{P}_\phi$ then becomes a $\mathcal{G}$-random measure, and the concentration bound holds conditionally on $\mathcal{G}$ with the same constants. We omit the measurability bookkeeping.*

B Proof of the Weighted Hitting-Time Bound

We prove the bound used in Section 4. Throughout this section, the normalized weights are fixed. Thus the result is weight-conditional: it analyzes the inner learning problem after the outer BOOST reweighting model has produced a weight vector.

Auxiliary result. We use the following range-length form of the PAC–Bayesian Hoeffding–Azuma inequality of Seldin et al. [2012].

Theorem B.1 (PAC–Bayesian Hoeffding–Azuma; Seldin et al., 2012)**.** *Let Π be a policy class and let P_0 be a prior distribution over Π chosen independently of the training sample. For each $\pi \in \Pi$, let $X_1(\pi), \dots, X_n(\pi)$ be martingale differences with respect to a filtration $\{\mathcal{F}_i\}_{i=0}^n$:*

$$\mathbb{E}[X_i(\pi) \mid \mathcal{F}_{i-1}] = 0.$$

Assume that, for each i, there is a deterministic constant $r_i \geq 0$ such that for every $\pi \in \Pi$, the random variable $X_i(\pi)$ is almost surely contained in an interval of length at most r_i. Then, for any $1 < c \leq e$, with probability at least $1 - \delta$, every posterior distribution Q over Π satisfies

$$\left| \mathbb{E}_{\pi \sim Q} \left[\sum_{i=1}^n X_i(\pi) \right] \right| \leq \frac{1+c}{2\sqrt{2}} \sqrt{\left(\mathrm{KL}(Q \| P_0) + \log \frac{2}{\delta} + \varepsilon_c(Q) \right) \sum_{i=1}^n r_i^2},$$

where

$$\varepsilon_c(Q) = \frac{\log 2}{2 \log c} \left[1 + \log \left(1 + \frac{\mathrm{KL}(Q \| P_0)}{\log(2/\delta)} \right) \right].$$

Corollary B.2 (Weighted PAC–Bayesian bound for bounded losses)**.** *Let $\ell(z, \pi) \in [0,1]$ be a bounded loss. Let $Z_1, \dots, Z_n$ be independent tasks with $Z_i \sim \mathcal{P}_i$. Fix weights $w \in \Delta_n$, where $w_i \geq 0$ and $\sum_{i=1}^n w_i = 1$. Define*

$$\widehat{L}_w(\pi) = \sum_{i=1}^n w_i \ell(Z_i, \pi), \qquad L_w(\pi) = \sum_{i=1}^n w_i \mathbb{E}_{Z \sim \mathcal{P}_i}[\ell(Z, \pi)],$$

and

$$n_{\mathrm{eff}}(w) = \frac{1}{\sum_{i=1}^n w_i^2}.$$

Then, for any $1 < c \leq e$, with probability at least $1 - \delta$, every posterior Q over Π satisfies

$$\mathbb{E}_{\pi \sim Q}[L_w(\pi)] \leq \mathbb{E}_{\pi \sim Q}[\widehat{L}_w(\pi)] + \frac{1+c}{2\sqrt{2}} \sqrt{\frac{\mathrm{KL}(Q \| P_0) + \log \frac{2}{\delta} + \varepsilon_c(Q)}{n_{\mathrm{eff}}(w)}}.$$

Proof. Fix $\pi \in \Pi$ and define

$$\mu_i(\pi) = \mathbb{E}_{Z \sim \mathcal{P}_i}[\ell(Z, \pi)].$$

Let

$$X_i(\pi) = w_i \left(\mu_i(\pi) - \ell(Z_i, \pi) \right).$$

Since Z_i is independent of $Z_1, \dots, Z_{i-1}$,

$$\mathbb{E}[X_i(\pi) \mid \mathcal{F}_{i-1}] = w_i \left(\mu_i(\pi) - \mathbb{E}[\ell(Z_i, \pi)] \right) = 0.$$

Thus $X_1(\pi), \dots, X_n(\pi)$ are martingale differences.

Since $\ell(Z_i, \pi) \in [0,1]$,

$$\mu_i(\pi) - \ell(Z_i, \pi) \in [\mu_i(\pi) - 1, \mu_i(\pi)].$$

Multiplying by $w_i \geq 0$ gives

$$X_i(\pi) \in [w_i(\mu_i(\pi) - 1), w_i \mu_i(\pi)].$$

This interval may depend on π, but its length is always w_i. Therefore the range-length parameter in Theorem B.1 is $r_i = w_i$, and

$$\sum_{i=1}^n r_i^2 = \sum_{i=1}^n w_i^2 = \frac{1}{n_{\mathrm{eff}}(w)}.$$

Finally,

$$\sum_{i=1}^{n} X_i(\pi) = L_w(\pi) - \widehat{L}_w(\pi).$$

Applying Theorem B.1 and using the upper-tail direction proves the result. □

Theorem B.3 (Exact Seldin-form weighted hitting-time bound)**.** *Assume $0 \le T_z^\pi \le H_{\max}$ for every task z and policy π. Fix normalized BOOST weights $\tilde{w}_{\phi,1}, \dots, \tilde{w}_{\phi,n}$ over $\mathcal{D}_{\text{mixed}}$, and let $\mathcal{P}_\phi$ be the weighted mixed task distribution defined by*

$$\mathbb{E}_{z\sim\mathcal{P}_\phi}[f(z)] = \sum_{i=1}^{n} \tilde{w}_{\phi,i}\, \mathbb{E}_{Z\sim\mathcal{P}_i}[f(Z)].$$

Let P_0 be the pretrained-policy prior chosen independently of the task-specific training trajectories, and let Q_θ be the posterior induced by the learned policy. Then, for any $1 < c \le e$, with probability at least $1-\delta$,

$$\mathbb{E}_{\pi\sim Q_\theta}\mathbb{E}_{z\sim\mathcal{P}_{\text{task}}}[T_z^\pi] \le \mathbb{E}_{\pi\sim Q_\theta}\left[\sum_{i=1}^{n} \tilde{w}_{\phi,i}\, T_{z_i}^\pi\right] + H_{\max}\frac{1+c}{2\sqrt{2}}\sqrt{\frac{\mathrm{KL}(Q_\theta\|P_0) + \log\frac{2}{\delta} + \varepsilon_c(Q_\theta)}{n_{\text{eff}}(\tilde{w}_\phi)}} + \Delta_{\text{task}}(\phi).$$

Proof. Set $\ell(z,\pi) = T_z^\pi / H_{\max}$; by the bounded-horizon assumption, $\ell(z,\pi) \in [0,1]$. Applying Corollary B.2 with $w_i = \tilde{w}_{\phi,i}$, $Q = Q_\theta$, and multiplying both sides by $H_{\max}$ yields

$$\mathbb{E}_{\pi\sim Q_\theta}\mathbb{E}_{z\sim\mathcal{P}_\phi}[T_z^\pi] \le \mathbb{E}_{\pi\sim Q_\theta}\left[\sum_{i=1}^{n} \tilde{w}_{\phi,i}\, T_{z_i}^\pi\right] + H_{\max}\frac{1+c}{2\sqrt{2}}\sqrt{\frac{\mathrm{KL}(Q_\theta\|P_0) + \log\frac{2}{\delta} + \varepsilon_c(Q_\theta)}{n_{\text{eff}}(\tilde{w}_\phi)}}.$$

By definition of $\Delta_{\text{task}}(\phi)$, for every $\pi \in \Pi$, $\mathbb{E}_{z\sim\mathcal{P}_{\text{task}}}[T_z^\pi] \le \mathbb{E}_{z\sim\mathcal{P}_\phi}[T_z^\pi] + \Delta_{\text{task}}(\phi)$. Taking expectation over $\pi \sim Q_\theta$ and combining the two inequalities proves the claim. □

Corollary B.4 (Simplified constant-form bound)**.** *Under the same assumptions as Theorem B.3, there exists a universal numerical constant $C_1 > 0$ such that, for any $\delta \in (0,1)$, with probability at least $1-\delta$,*

$$\mathbb{E}_{\pi\sim Q_\theta}\mathbb{E}_{z\sim\mathcal{P}_{\text{task}}}[T_z^\pi] \le \mathbb{E}_{\pi\sim Q_\theta}\left[\sum_{i=1}^{n} \tilde{w}_{\phi,i} T_{z_i}^\pi\right] + C_1 H_{\max}\sqrt{\frac{\mathrm{KL}(Q_\theta\|P_0) + \log(2/\delta)}{n_{\text{eff}}(\tilde{w}_\phi)}} + \Delta_{\text{task}}(\phi).$$

Proof. Fix any $1 < c \le e$ in Theorem B.3. Since $\log(1+x) \le x$ for $x \ge 0$, the term $\varepsilon_c(Q_\theta)$ is bounded by a constant depending only on c times $\mathrm{KL}(Q_\theta\|P_0) + \log(2/\delta)$. Absorbing this constant and the Hoeffding–Azuma prefactor $(1+c)/(2\sqrt{2})$ into C_1 proves the claim. □

C Hyperparameters

In Table 3, we provide the hyperparameter details for all our experiments.

D Task Categories and Category-Based Data Splits

A central design choice in our evaluation is to split data *by task category* rather than by individual task instance. This tests genuine generalisation: a model must transfer its learned policy to semantic categories it has never optimised on, rather than simply interpolating within a known distribution. Below we describe the categories for each environment and the precise procedure used to construct the splits.

Table 3: Hyperparameters for all experiments across three environments and two algorithms.

Environment	Algo.	Model	LR	α (init)	CQL weight	Batch size	Grad. accum.	$K_\theta = K_\psi$	K_ϕ	Train frac	Val split
Twenty Questions	MC	`gpt2`	1e-4	0	10	8	16	20	1	0.6	0.3
Twenty Questions	ILQL	`gpt2`	1e-4	1	10	8	16	20	1	0.6	0.3
Car Dealer	ILQL	`gpt2`	1e-4	1	10	8	2	32	1	0.6	0.3
Car Dealer	MC	`gpt2`	1e-4	10	10	8	2	32	1	0.6	0.3
WebShop	MC	`LLaMA-3-8B`	1e-4	0.1	10	4	4	32	1	0.6	0.3
WebShop	ILQL	`LLaMA-3-8B`	1e-4	0.1	10	1	16	32	1	0.6	0.3

D.1 Twenty Questions: Word Categories

In the Twenty Questions environment each episode contains a hidden target word that the agent must identify through yes/no questions. Every target word belongs to one of 17 semantic categories. The full category taxonomy with representative members is given in Table 4.

Table 4: Twenty Questions word categories and sample objects. The full item pool contains 10–14 objects per category.

Category	Sample objects
Sports	Basketball, Football, Baseball, Tennis racket, Helmet
Animals	Cat, Dog, Horse, Lion, Elephant
Fruits	Apple, Banana, Strawberry, Watermelon, Mango
Vehicles	Car, Motorcycle, Airplane, Helicopter, Ship
Clothes	Shirt, Jacket, Dress, Boots, Scarf
Electronics	Smartphone, Television, Camera, Refrigerator, Blender
Musical Instruments	Piano, Guitar, Violin, Trumpet, Harp
Furniture	Chair, Bed, Couch, Bookcase, Nightstand
Office Supplies	Pen, Stapler, Calculator, Scissors, Diary
Vegetables	Carrot, Broccoli, Tomato, Spinach, Cucumber
Art	Painting, Paintbrush, Canvas, Sculpture, Marker
Kitchen Tools	Knife, Fork, Bowl, Frying pan, Whisk
Nature	Rock, Tree, Mountain, Ocean, Cactus
Toys	Lego, Doll, Kite, Puzzle, Stuffed animal
Jewelry	Earrings, Necklace, Ring, Watch, Pendant
Garden Supplies	Shovel, Rake, Watering can, Lawn mower, Gloves
Tools	Hammer, Screwdriver, Wrench, Saw, Drill

D.2 Car Dealer Negotiation: Brand Categories

In the Car Dealer Negotiation environment the buyer's preferred car brand encodes latent information about price range, feature expectations, and negotiation style. We treat each brand as a distinct task category. The 13 brands used in our experiments are:

Volkswagen, Lexus, Ford, Mazda, Hyundai, Toyota, Mercedes-Benz, BMW, Audi, Subaru, Honda, Porsche, Tesla.

D.3 WebShop Agent: Product Categories

Each WebShop episode specifies a purchase goal tied to a product drawn from Amazon's category taxonomy. We use the full two-level category string as the task category label. Our dataset spans 37 such categories, listed in Table 5.

Table 5: WebShop product categories used as task categories (37 total).

Category (domain – sub-category)	
Automotive – Motorcycle & Powersports	Baby Products – Nursery
Beauty & Personal Care – Foot, Hand & Nail Care	Beauty & Personal Care – Fragrance
Beauty & Personal Care – Hair Care	Beauty & Personal Care – Makeup
Beauty & Personal Care – Oral Care	Beauty & Personal Care – Personal Care
Beauty & Personal Care – Shave & Hair Removal	Beauty & Personal Care – Skin Care
Beauty & Personal Care – Tools & Accessories	Cell Phones & Accessories – Accessories
Cell Phones & Accessories – Cases, Holsters & Sleeves	Clothing, Shoes & Jewelry – Men
Clothing, Shoes & Jewelry – Novelty & More	Clothing, Shoes & Jewelry – Sport Specific Clothing
Clothing, Shoes & Jewelry – Women	Electronics – Accessories & Supplies
Electronics – Camera & Photo	Electronics – Car & Vehicle Electronics
Electronics – Computers & Accessories	Electronics – Home Audio
Electronics – Portable Audio & Video	Grocery & Gourmet Food – Deli & Prepared Foods
Grocery & Gourmet Food – Frozen	Grocery & Gourmet Food – Produce
Grocery & Gourmet Food – Snacks & Sweets	Home & Kitchen – Bedding
Home & Kitchen – Furniture	Home & Kitchen – Home Décor Products
Home & Kitchen – Kitchen & Dining	Home & Kitchen – Wall Art
Office Products – Office & School Supplies	Office Products – Office Furniture & Lighting
Sports & Outdoors – Sports	Tools & Home Improvement – Lighting & Ceiling Fans

D.4 Shared Split Protocol

Across all three environments the split procedure follows these steps:

1. **Category enumeration.** All distinct category labels present in the full dataset are collected and fixed before any model training.
2. **Train/eval category partition.** A fraction f_{train} of categories (controlled by `-train_task_frac`) is randomly assigned to the *training* split. All remaining categories form the *eval category pool*, which is shared between the validation and evaluation sets. All offline trajectories whose category falls in the training split are placed in $\mathcal{D}_{\text{train}}$.
3. **Val/eval trajectory partition within held-out categories.** For every category in the eval category pool, its trajectories are randomly divided into two groups. A fraction f_{val} of trajectories (controlled by `-val_split`) are assigned to the *validation* set $\mathcal{D}_{\text{val}}$; the remaining $1 - f_{\text{val}}$ fraction form the *evaluation* set $\mathcal{D}_{\text{eval}}$. Crucially, validation and evaluation trajectories are drawn from the *same held-out categories*.
4. **Synthetic data generation.** Synthetic trajectories are seeded exclusively from prefixes of $\mathcal{D}_{\text{train}}$ trajectories. No synthetic data is generated from held-out categories, ensuring that $\mathcal{D}_{\text{synth}}$ shares the category distribution of $\mathcal{D}_{\text{train}}$ and leaks no information about held-out categories into training.
5. **Evaluation.** At test time, the policy interacts with a live simulator on tasks drawn from the held-out categories exclusively, using the $\mathcal{D}_{\text{eval}}$ partition. This constitutes a strictly out-of-distribution evaluation at the *category level*.

6. **Low data sample.** To simulate a low-data setting, we subsample 2.5%, 10%, and 30% of the full training data for Twenty Questions, Car Dealer, and WebShop respectively.

The exact split fractions used per environment are summarised in Table 3. The partition is fixed with a single random seed before any model training and held constant across all methods for a given environment, ensuring fair comparison.

E Synthetic Data Generation

For each training trajectory $\tau \in \mathcal{D}_{\text{train}}$ we generate exactly one synthetic counterpart, so $|\mathcal{D}_{\text{synth}}| = |\mathcal{D}_{\text{train}}|$. To ensure a fair comparison with non-synthetic baselines, we always use the *same* LLM that will subsequently be fine-tuned as the data generator: GPT-2 for Twenty Questions and Car Dealer Negotiation, and Llama-3-8B for WebShop. This avoids any confound from generator capacity: synthetic baselines cannot benefit from access to a stronger model, so performance differences are attributable solely to the reweighting mechanism.

E.1 Generation Procedure

Each synthetic trajectory is constructed as follows.

1. **Seeding from real data.** We take the first two timesteps of a real training trajectory, one environment observation and one agent action, as the prefix of the synthetic trajectory. This grounds the synthetic episode in the same initial conditions as the real data and ensures that synthetic tasks are drawn from the same category distribution as $\mathcal{D}_{\text{train}}$.
2. **Simulator-free self-play.** From the seed prefix onward, the LLM is prompted to play *both* roles simultaneously: it generates the next agent action conditioned on the current history, then generates the resulting environment observation conditioned on that action, and so on. Crucially, this requires no access to the real environment simulator. At each step, the model receives (i) a system prompt describing the environment rules and action space, (ii) an example of a valid trajectory, and (iii) the full dialogue history so far.
3. **Termination.** The episode ends when the model generates a terminal action (e.g. `click[Buy Now]` in WebShop, a final guess in Twenty Questions, or an agreement or rejection in Car Dealer) or when the maximum episode length is reached.

This procedure yields two important properties. First, because the same model acts as both agent and environment, the synthetic observations are consistent with the model's own internal world model, keeping trajectories internally coherent without a simulator. Second, because synthetic data is seeded from real prefixes, the early timesteps of each synthetic trajectory remain grounded; distributional drift accumulates only in later turns, where the bilevel reweighting head can learn to discount low-quality continuations.

E.2 Environment Prompts

Below, in Figures 4, 5, 6 we report the prompts used to condition the LLM in each role during synthetic data generation for Twenty Questions, Car Dealer and Webshop Environments.

In all three environments, a representative example trajectory (formatted identically to the real data) is appended to the respective system prompts before generation begins, to disambiguate the expected output format and action grammar.

F Training Implementation Details

F.1 Base Language Models

For Twenty Questions and Car Dealer Negotiation we fine-tune **GPT-2** (117M parameters), which has been shown to perform adequately on these tasks in prior work Abdulhai et al. [2025]. For WebShop, which requires multi-step information gathering and diverse action types across a large product space, we fine-tune **Llama-3-8B** (8B parameters). To make fine-tuning tractable at this scale, we apply

Agent main prompt

```
Welcome to the game of Twenty Questions!  Your objective is to guess what
the object is within twenty questions.  At every turn, you will have the
opportunity to ask a yes/no question and receive an answer from the oracle.
You can ask twenty questions but must ask as few questions as possible.
```

Environment system prompt (oracle)

```
You are the oracle in a game of Twenty Questions.  A hidden target object has
been chosen (provided in the goal).  For every yes/no question the agent asks,
respond with exactly ‘‘Yes.’’  or ‘‘No.’’  based solely on whether the answer is
true of the target object.  Do not reveal the object or provide any other
information.  Output only your one-word answer.
```

Figure 4: Twenty Questions prompts used during synthetic data generation.

Agent system prompt (salesperson)

```
You are an expert car salesperson conducting a price negotiation with a buyer.
Your goal is to reach a sale at the highest possible price while keeping the
buyer engaged.  You may offer discounts and highlight vehicle features, but
you must not accept a price below your minimum threshold.  Output only a single
dialogue turn.
```

Environment system prompt (buyer)

```
You are simulating a car buyer in a negotiation.  Given the negotiation history,
generate the buyer’s next response.  The buyer has a target price and preferred
brand (provided in the goal).  The buyer will counteroffer when the current
price exceeds their target, accept if it meets or falls below their target, and
walk away after a fixed number of failed rounds.  Output only the buyer’s next
utterance, nothing else.
```

Figure 5: Car Dealer Negotiation prompts used during synthetic data generation. Bracketed placeholders are filled from the seed trajectory's metadata.

QLoRA Dettmers et al. [2023]: the base model weights are quantised to 4-bit precision and kept frozen, while low-rank adapter matrices are trained in 16-bit precision. This reduces the memory footprint of the base model significantly.

F.2 Q and V Head Architecture

BOOST augments the base language model with lightweight value heads that are trained on top of base LLMs final hidden layer outputs. The head architecture differs by RL algorithm:

- **Monte Carlo (MC).** A single Q-head maps each token's hidden state to a scalar return estimate. The head is a linear projection:

$$Q_\theta(s, a) \;=\; W_Q\, h \;+\; b_Q, \quad W_Q \in \mathbb{R}^{|\mathcal{V}| \times d},$$

  where $h \in \mathbb{R}^d$ is the final hidden state of the LM (with d the model embedding dimension) and $|\mathcal{V}|$ is the vocabulary size.

- **ILQL.** Two Q-heads (for the double-Q trick) and one V-head are used. The Q-heads share the same architecture as in MC above; the V-head is a scalar projection:

$$V_\theta(s) \;=\; W_V\, h \;+\; b_V, \quad W_V \in \mathbb{R}^{1 \times d}.$$

Both head types are thus *depth-1 linear models* with no non-linearity, keeping the number of additional parameters negligible relative to the base LM.

Agent system prompt

```
You are playing WebShop.  You will see a text view of the current page and must
output exactly one action per turn.

CRITICAL RULES:
- search[query]:  type a search query of 4-8 words maximum, extracting only
core keywords (e.g., ‘‘mens tuxedo shirt’’).
- You only get ONE search.  Once you see search results, choose the
best-matching product and use click[].
- On a product page, click the relevant attribute options and then click[Buy
Now].
- If attributes or price do not match and a better item exists, click[< Prev]
to return to results and try the next item.

Output only one action per message in the form action[argument].  Nothing
else.
```

Environment simulator system prompt

```
You are simulating the WebShop text-based shopping environment.  Given the
trajectory so far, generate the NEXT environment observation.

RULES:
1.  After search[query]:  return 3 product listings in the format
  [ASIN] \n Product Name \n $Price
Include the target product (from goal info) as one of the 3 results.
2.  After click[ASIN] on results:  return the product detail page with option
dropdowns, product name, price, rating, and [Buy Now].  Use real option values
for the target product.
3.  After click[option] on a product page:  return ‘‘You have clicked {option}.’’
4.  After click[< Prev]:  return the most recent search results page.

Output ONLY the observation text.  Match exact WebShop formatting.
```

Figure 6: WebShop prompts used during synthetic data generation.

F.3 Memory-Efficient Auxiliary Model

BOOST maintains an auxiliary policy ψ that tracks the inner-level optimum (see Algorithm 1). Instead of storing a full second copy of all model parameters, doubling memory consumption our auxiliary policy ψ and the primary policy θ differ only in their value heads: the base LM backbone is shared between the two. Concretely, we keep a *single copy* of the base LM weights in GPU memory and maintain *separate copies of only the Q-heads* (for MC) or *the Q-heads and V-head* (for ILQL) for θ and ψ respectively. Because the value heads are depth-1 linear layers, the additional memory cost of the auxiliary model is a negligible fraction of the total.

F.4 Reweighting Head w_ϕ

The reweighting head w_ϕ takes as input an embedding representation of a trajectory and outputs a scalar weight. We use a fully connected MLP with ReLU activations:

- **Twenty Questions and Car Dealer Negotiation:** depth-2 MLP (one hidden layer).
- **WebShop:** depth-4 MLP (three hidden layers), reflecting the greater complexity and diversity of the WebShop action space.

In all cases the MLP is small relative to the base model, adding negligible compute and memory overhead to the bilevel updates.

F.5 Trajectory Embeddings

To obtain a fixed-dimensional representation of each trajectory for input to w_ϕ, we pass the trajectory through the *pretrained* (unmodified) base language model and extract the final-layer hidden state of

the last token as the trajectory embedding. This embedding is computed once per trajectory before training begins and cached, so it does not contribute to the per-iteration training cost. Using the pretrained model's representations rather than the continuously updating fine-tuned model ensures that the embedding space remains stable throughout training, preventing the input distribution to w_ϕ from shifting as θ evolves.

G Ablation of Stabilizing Design Choices

Table 6: $2 \times 2 \times 2$ ablation of the design choices in Section 3.3 (WebShop, MC, low-data). Reward is mean $\pm$ standard error. g_{rel} is the relative loss gap between the primary policy θ and the auxiliary policy ψ (lower is better).

Arm	Phased updates	Warm-start ($\theta \leftarrow \psi$)	Slow LR (ϕ)	Reward $\uparrow$	$g_{\text{rel}} \downarrow$
1	✓	✓	✓	$\mathbf{0.433 \pm 0.031}$	$\mathbf{0.005}$
2	✓	✓	–	0.405 ± 0.034	0.012
3	✓	–	✓	0.269 ± 0.048	0.445
4	✓	–	–	0.209 ± 0.081	0.463
5	–	✓	✓	0.207 ± 0.145	0.106
6	–	✓	–	0.365 ± 0.045	0.096
7	–	–	✓	0.233 ± 0.046	1.941
8	–	–	–	0.196 ± 0.041	1.895

To assess how much each design choice in Section 3.3 contributes to stability, we run a full $2 \times 2 \times 2$ ablation over: (i) phased (alternating) updates of ψ, θ, and ϕ; (ii) warm-starting the primary policy from the auxiliary policy ($\theta \leftarrow \psi$) at each outer iteration; and (iii) a slow learning rate for the reweighting head ϕ (asymmetric learning rates).

We define the relative loss gap as $g_{\text{rel}} = |\mathcal{L}_{\text{train}}(\theta, \phi) - \mathcal{L}_{\text{train}}(\psi, \phi)| \,/\, \mathcal{L}_{\text{train}}(\psi, \phi)$, which measures how closely the primary policy θ tracks the inner-level optimum ψ. A small g_{rel} is critical for the bilevel formulation to be a good approximation, since we rely on the assumption $\mathcal{L}_{\text{train}}(\theta, \phi) \approx \mathcal{L}_{\text{train}}(\psi, \phi)$, i.e., the primary policy stays close to the inner-level optimum under the learned weights. Across Arms 1–8 (Table 6), we observe:

- **Phased updates and warm-start each independently help.** Turning on either one reduces the loss gap relative to the fully-off configuration (Arm 8). When both are on (Arms 1–2), the gap shrinks by more than two orders of magnitude (0.005–0.012 vs. 1.895) and reward is highest, showing that both choices are needed for bilevel optimization to behave as intended in practice .
- **The slow learning rate for ϕ plays a secondary role.** It is most useful together with phased updates (Arm 1 vs. 2: 0.433 vs. 0.405, with the gap halved. In settings without other stabilising tricks (Arm 7 vs 8), it modestly improves reward while keeping the gap from further exploding.

These results show that each design choice contributes to stable and effective trajectory-level bilevel reweighting.

H Compute Infrastructure

We run all our experiments on Nvidia A100 GPUs with 80 GB memory. Each experiment on Twenty Question environment runs within 14 hours while experiments on Car Dealer Environment and Webshop environment run within 10 hours.

I Qualitative Analysis of Synthetic Data

Table 7 provides a side-by-side comparison of WebShop synthetic trajectories assigned high and low weights by w_ϕ. The high-weight trajectory follows the task instruction faithfully, executes actions that are grounded in environment observations, and completes a successful purchase. The low-weight trajectory, by contrast, exhibits two characteristic failure patterns: hallucinated action targets, clicking on a product identifier that never appeared in the environment's output, and environment-rule

Table 7: Side-by-Side Comparison of Good vs. Bad WebShop Trajectories

Phase / Key Timestep	**Good Synthetic Sample (%weight shift: 5.2%)**	**Bad Synthetic Sample % weight shift:-2.8%**
T0: Init	**State:** Environment reset; instruction specifies long-sleeve men's T-shirt, color *green*, size *3x-large big*, price $< \$70$. **Action:** (implicit) `reset` **Reward:** 0.0	**State:** Environment reset; instruction with dress shirt, specific fabrics, color *kelly green*, fit *men*, size *2T*, price $< \$50$. **Action:** (implicit) `reset` **Reward:** 0.0
T1: Search	**State:** Main webpage with instruction. **Action:** `search[machine wash men's t-shirts long sleeve green 3x-large]` **Reward:** 0.0 **Comment:** Single query with key attributes (category, sleeve, color, size).	**State:** Main webpage with instruction. **Action:** `search[mens dress shirts kelly green 2t]` **Reward:** 0.0 **Comment:** single query; aligned with the goal at this point.
T2: First Product Choice	**State:** Search results show multiple long-sleeve T-shirts, including `B00O30JLDK` with relevant attributes. **Action:** `click[B00O30JLDK]` **Reward:** 0.0 **Comment:** Chooses a *valid* ID from the current results (no hallucination).	**State:** Search results show IDs `B0969G2DH8`, `B07F9K8K3R`, `B07R9T8K3R`. **Action:** `click[B09QQP3356]` **Reward:** 0.0 **Comment:** Hallucinates an ID not in results, causing jump to unrelated product.
T3: Refinement vs. Drift	**State:** On product page `B00O30JLDK`; size options include *3x-large big*, color options include *green*. **Action:** `click[3x-large big]` **Reward:** 0.0 **Comment:** Refines selection by choosing the exact requested size; still on a good candidate item.	**State:** On a polo shirt page (`B09QQP3356`) with wrong category (polo, short sleeve), wrong colors, no 2T. **Action:** `think[no valid action]` and then attempts a second search later. **Reward:** 0.0 **Comment:** Recognizes difficulty but fails to recover (should navigate back or pick a valid ID from original results).
T4: Near-Term Goal Alignment	**State:** Same product page, now with size set correctly; price $\$49.99 < \$70$, color *green* available. **Action:** `click[Buy Now]` **Reward: 1.0** **Comment:** All constraints (type, sleeve, color, size, price) satisfied at checkout.	**State:** After bouncing between unrelated items and invalid actions, no product matches color *kelly green* and size *2T*. **Action:** `click[Buy Now]` from a mismatched context. **Reward: 0.0** **Comment:** Attempts to check out without satisfying core constraints; failure stems from early hallucination plus rule violations and lack of recovery.

violations in the form of a forbidden repeated search, both of which derail task completion before a reward can be obtained. That w_ϕ learns to distinguish these cases without explicit supervision on action validity is noteworthy: the reweighting signal comes purely from downstream validation reward, yet the learned weights recover a semantically meaningful notion of trajectory quality. This qualitative evidence is consistent with our quantitative findings in Figure 2 that BOOST assigns higher weights to synthetic transitions that most closely resemble valid, executable offline behaviour.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the three main contributions:trajectory level bilevel reweighting with synthetic data, PAC-Bayesian Analysis of generalization performance, and empirical improvements and analysis of results.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We briefly mention the limitation of our work in the conclusion, specifying that we only increase reweighting granularity upto trajectory level and a natural extension would be to learn weights at per dialogue level.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes],

Justification: We provide a thorough set of assumptions of our theoretical results and proof, particularly the setting of finite horizon setting .

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: In the appendix, we give complete details of hyperparameters used in all our experiment results. We also give details on the BOOST algorithm framework along with the training setup, data generations setup and prompts used for synthetic data generation.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provide a zip file of the codebase for running the experiments.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide the core details of experimental setup int he main paper. In the appendix, we give complete set of hyperparameters as well as data generation and data split details.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Yes, we provide both mean and standard error in the main results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer:[Yes]

   Justification: In the appendix, we provide details on the compute infrastructure used and the summary of compute time.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Yes, the research adheres to the Code od Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This work is situated in a broad class of works on fine-tuning LLMs and the framework proposed does not directly have any Societal impact.

    Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: No new models or datasets are released.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [N/A]

    Justification: No licenced assets are used. Prior works are cited wherever applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: No new assets are released in this paper.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve research with human subjects or any crowdsourcing.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve research with human subjects or any crowdsourcing.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLM is not used in core development in this paper.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.