Reward Shaping to Improve Language Model Query Generation

Shicheng Liu[1], **Zeyu Zhang**[2], **Lei Lu**[3], **Kexuan Sun**[2], **Thuy Vu**[2]
[1]MIT CSAIL [2]Amazon AGI [3]Northeastern University ECE
liushch@mit.edu, {zeyzhan,sunkexuan,thuyvu}@amazon.com,
lu.lei1@northeastern.edu

Abstract

Language model query generation is the task of using a language model to translate a natural-language request into a formal query, which is then executed to retrieve information from databases or knowledge graphs. Recent approaches rely on reinforcement learning (RL) to enhance the model's reasoning and improve query correctness. However, existing methods provide only sequence-level rewards, resulting in sparse feedback that is difficult to optimize. To address this limitation, we propose a reward shaping method that learns a token-level, and thus dense, reward function, providing fine-grained supervision to guide the language model toward correct queries. We formulate this problem as a bilevel optimization problem where the upper level learns a token-level shaping reward from ground truth sequence-level reward and the lower level optimizes the language model using the learned shaping reward. We propose a single-loop algorithm to solve this problem and theoretically guarantee that the algorithm converges at $O(\frac{1}{\sqrt{K}}) + O(\frac{\log K}{\sqrt{K}})$. Extensive experiments demonstrate that our method outperforms the baselines.

1 Introduction

In modern information retrieval systems, such as search engines, user requests are typically expressed in natural language, whereas the underlying systems operate over structured query languages such as SQL and Cypher. Consequently, a key step in information retrieval is language model query generation, which leverages a language model to translate natural-language requests into queries that can be executed to retrieve relevant information from databases or knowledge graphs. Language model query generation has been widely adopted in industry, including Google BigQuery [1], Microsoft Power BI [2], and Amazon QuickSight [3].

Despite its importance, language model query generation is a challenging task that requires natural language understanding, database schemas or knowledge graph ontologies comprehension, and precise query formulation. Recent approaches [4, 5] have employed reinforcement learning (RL) to enhance the reasoning capability of language models and improve query generation correctness. These methods typically design composite reward functions to supervise language model optimization. Such composite rewards generally consist of two components: a correctness reward and one or more partial rewards. The correctness reward is a binary signal indicating whether the generated query retrieves the correct results. However, this binary reward is difficult to optimize, as it assigns identical feedback to queries with different degrees of correctness, such as nearly correct queries and completely incorrect ones. To address this issue, partial rewards are introduced to provide more fine-grained supervision [4], including format rewards that evaluate response structure, syntax rewards that check syntactic validity, and length rewards that encourage sufficiently long and well-structured reasoning.

Although composite rewards provide richer feedback than pure correctness rewards, they suffer from two fundamental limitations. First, both correctness and composite rewards are sequence-

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

level rewards and thus provide sparse feedback. In reinforcement learning [6], a reward function is considered sparse if it assigns rewards only at the terminal state–action pair of a trajectory (e.g., terminal token of a sequence), regardless of the magnitude of the reward. Sparse rewards are typically hard to optimize (elaborated in Section 3.2). Second, partial rewards may introduce misaligned learning signals. While the ultimate objective of query generation is to produce correct queries, partial rewards emphasize auxiliary properties such as formatting quality, syntactic validity, and reasoning length that do not necessarily correspond to query correctness. Since optimizing these properties is often easier than ensuring query correctness, partial rewards may lead to reward hacking, where the language model generates well-structured and syntactically valid responses but the queries are incorrect.

To address these two issues, we propose a reward shaping method that learns a shaping reward for language model query generation. The shaping reward has two features: (i) it provides token-level feedback, enabling fine-grained credit assignment and easier optimization; (ii) it is explicitly optimized for alignment with the correctness objective, by learning the shaping reward according to the correctness achieved by its downstream policy.

Contribution statement. Our contributions are threefold.

First, we introduce a principled framework for learning token-level shaping rewards for language model query generation. Motivated by potential-based reward shaping, we design a token-level reward parameterization that provides dense supervision for policy optimization. Rather than relying on the reward parameterization alone to ensure alignment, we formulate a bilevel optimization problem where the upper level learns the shaping reward to maximize the correctness reward achieved by the downstream policy, while the lower level optimizes the policy using the learned shaping reward.

Second, we propose an efficient single-loop algorithm, termed **Re**ward **shap**ing for qu**e**ry generation (RESHAPE), that jointly updates the shaping reward and the language model policy via one-step updates at each iteration. In contrast to standard double-loop algorithms that repeatedly solve the lower-level policy optimization problem to near optimality before updating the upper-level shaping reward, our algorithm avoids nested optimization and thus significantly reduces per-iteration computational cost.

Third, we formally guarantee that RESHAPE converges at a rate of $O(\frac{1}{\sqrt{K}}) + O(\frac{\log K}{\sqrt{K}})$. A key technical component of our analysis is the use of two-timescale stochastic approximation to stabilize the lower-level approximation error induced by the one-step policy update and guarantee the overall convergence. Experiments on SQL query generation and Cypher query generation demonstrate that our method outperforms state-of-the-art RL methods.

2 Related works

Language model query generation. Early works on language model query generation primarily focus on zero-shot and few-shot learning via prompt engineering and in-context learning [7, 8, 9]. More recent approaches fine-tune language models on text-to-query datasets to improve performance [10, 11, 12, 13]. Our previous work on web-based question answering [14, 15, 16, 17, 18, 19] also studies in this setting. However, supervised fine-tuning requires golden queries, which are often expensive and time-consuming to obtain at scale. Recently, [20] proposes optimizing language models using the correctness reward. Building on this idea, subsequent works [4, 5] introduce composite reward functions that combine the correctness rewards with partial rewards to provide richer supervision signals. Despite their effectiveness, both correctness rewards and composite rewards are sequence-level and only provide sparse feedback. In addition, partial rewards may introduce bias by encouraging auxiliary properties that are not fully aligned with the ultimate objective of generating correct queries. In contrast, this paper learns a token-level shaping reward that provides dense, fine-grained supervision while being explicitly optimized according to the downstream correctness objective.

Reward shaping. Reward shaping can improve the RL performance by shaping the ground truth reward function. Existing reward shaping methods can be broadly categorized into rule-based and optimization-based approaches. Rule-based methods design shaping rewards using predefined heuristics, such as potential-based shaping [21, 22], distance-based shaping [23, 24], and subgoal-based shaping [25, 26]. Optimization-based methods [27, 28, 29] instead learn shaping rewards from

the ground truth reward signal. Among them, [30, 31] are most closely related to our work, as they learn shaping rewards from binary correctness reward signals using a cross-entropy loss. In contrast, we formulate a bilevel optimization problem, in which the shaping reward is learned at the upper level with the explicit objective that the corresponding policy maximizes the correctness reward.

Bilevel optimization. Algorithms for bi-level optimization can be categorized into double-loop and single-loop methods. Double-loop methods [32, 33] follow a nested optimization structure that mirrors the hierarchical nature of the bi-level problem. In these approaches, the lower-level problem is (approximately) solved to optimality for a given upper-level variable, and the resulting solution is then used to update the upper-level objective. This process is repeated iteratively, with an inner loop devoted to solving the lower-level problem and an outer loop updating the upper-level variables. Double-loop methods are conceptually simple and align closely with the mathematical definition of bi-level optimization, but can be computationally expensive due to the repeated solution of the lower-level problem. Single-loop methods [34, 35] interleave the updates of the upper-level and lower-level variables within a single optimization loop. Rather than fully solving the lower-level problem at each iteration, these methods rely on implicit differentiation, truncated optimization, or gradient-based approximations to propagate lower-level information to the upper level. Single-loop methods are more computationally efficient, but require careful analysis to ensure convergence.

3 Preliminaries

In this section, we first model causal language model generation as a Markov decision process (MDP) in Subsection 3.1 and then elaborate why token-level rewards are helpful for the query generation task in Subsection 3.2.

3.1 Markov decision process

A (finite-horizon) Markov decision process (MDP) is a tuple $(\mathcal{S}, \mathcal{A}, P, P_0, T, \gamma, r)$. The state space is denoted by $\mathcal{S}$ where each state $s_t = (x, y_{0:t-1})$ includes the prompt x and all response tokens generated up to the latest token y_{t-1}. The initial state s_0 only includes the prompt x. The action space $\mathcal{A}$ is the vocabulary and each action $a_t = y_t$ represents a token y_t from the vocabulary. The state transition function $P(s_{t+1}|s_t, a_t)$ is deterministic and provides the next state $s_{t+1} = (x, y_{0:t})$ given the current state $s_t = (x, y_{0:t-1})$ and current action $a_t = y_t$. The initial state distribution P_0 corresponds to sampling a prompt x from a dataset $\mathcal{D}$, which we denote by $x \sim \mathcal{D}$. The time horizon T denotes the maximum number of tokens that can be generated and $\gamma \in [0, 1]$ is the discount factor.

The reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ assigns a scalar value to a state-action pair. A sparse sequence-level reward assigns zero to all intermediate state-action pairs and provides a scalar reward only at the terminal state-action pair:

$$r(s_t, a_t) = 0 \ \text{ for } 0 \le t < T, \quad r(s_T, a_T) \in \mathbb{R}.$$

Therefore, both the correctness reward and composite reward are sparse sequence-level rewards. In contrast, a dense token-level reward function assigns scalar values $r(s_t, a_t)$ to each state-action pair (s_t, a_t), providing intermediate feedback throughout the trajectory/sequence.

Remark on the equivalency between state-dependent and state-action-dependent reward functions. In our formulation, the reward function is defined as state–action dependent. However, this is equivalent to a state-dependent reward formulation. Since the state transition is deterministic and satisfies $s_{t+1} = (s_t, a_t)$, any reward function that depends on both the state and action can be equivalently expressed as a state-dependent reward on the next state, e.g., $r(s_t, a_t) = r(s_{t+1})$.

A language model can be seen as a stochastic policy π where $\pi(a_t|s_t)$ denotes the probability of generating the next token $a_t = y_t$ given the previous tokens $s_t = (x, y_{0:t-1})$.

3.2 Token-level reward helps query generation

In this subsection, we justify why token-level rewards are especially beneficial for query generation tasks. This stems from a fundamental distinction between query generation and standard natural language generation. In natural language generation, responses are semantically robust to small local errors. Minor typos or punctuation errors often do not significantly affect the overall meaning or

usability of the generated text. For example, while the grammatically correct sentence is "the weather is sunny," the sentence "the weather are sunny" remains easily understandable. In contrast, query generation tasks are highly brittle: even a single incorrect token, such as a misplaced syntax symbol or keyword, can render the entire query invalid.

Figure 1 illustrates this issue through four representative scenarios. The first query is syntactically and semantically correct and thus receives a correctness reward of $+1$. The second query differs from the first by only a single incorrect syntax symbol; however, it is entirely invalid and receives a correctness reward of 0. Although the error is minimal, the sequence-level correctness reward penalizes the entire query. Recent works [4, 5] attempt to mitigate this issue by introducing composite rewards that evaluate auxiliary properties such as syntax validity in addition to correctness. Under such a composite reward, the third query receives a larger negative reward (e.g., -3) due to both incorrect execution and syntax errors. However, despite incorporating additional signals, the composite reward remains sequence-level and still penalizes the entire query.

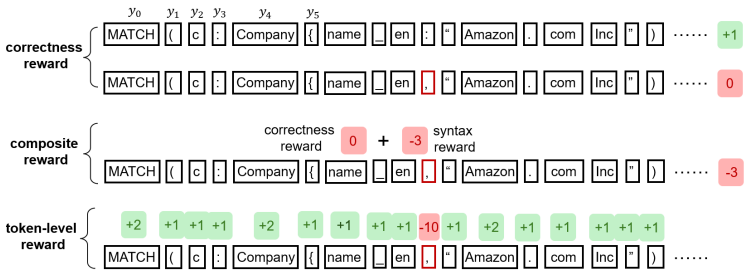

Figure 1: Four scenarios. The first query is correct and receives a correctness reward of +1. The second query is incorrect and receives a correctness reward of 0. The third query receives a composite reward of -3. The fourth query receives token-level rewards and only the wrong syntax symbol is penalized.

Under both correctness and composite rewards, the learning signal does not indicate where the error occurs, making it difficult for the language model to identify which tokens should be corrected. In contrast, a token-level reward assigns feedback to each token in the fourth query, penalizing only the incorrect tokens while preserving positive feedback for correct tokens. This fine-grained credit assignment enables the language model to more precisely localize and correct erroneous tokens, thereby making token-level rewards particularly effective for query generation tasks.

4 Problem formulation

In this section, we formulate a bilevel optimization problem where the upper level learns a token-level shaping reward and the lower level optimizes the language model using the learned shaping reward. This section includes two subsections where Subsection 4.1 discusses the parameterization of the shaping reward and Subsection 4.2 introduces the bilevel optimization problem.

4.1 Shaping reward parameterization

While token-level rewards provide fine-grained credit assignment and facilitate language model optimization, manually designing token-level reward functions based on heuristic rules requires substantial domain expertise and may introduce learning signals that are misaligned with the correctness objective. In query generation, the ultimate objective is to generate correct queries, which is directly captured by the sequence-level correctness reward. However, it is difficult and often infeasible to manually construct a token-level reward function that accurately reflects the global correctness of a query at each intermediate token.

We therefore aim to learn a dense token-level shaping reward whose parameters can be optimized according to the downstream correctness objective. To enable fine-grained credit assignment, we adopt a reward parameterization motivated by potential-based reward shaping. Importantly, our parameterization is not restricted to the classical policy-invariant potential-based form; instead, its alignment with the correctness objective is explicitly pursued through the upper-level optimization introduced in Section 4.2.

Toward this end, we first need to specify how to parameterize the shaping reward function. A common approach is to parameterize reward functions by adding a linear head on top of the final transformer

layer. However, such generic parameterizations do not provide any guarantee that optimizing the learned shaping reward will induce the same optimal policy as optimizing the correctness reward.

Potential-based reward shaping [21] provides useful motivation for constructing dense intermediate rewards. In classical potential-based shaping, the original $r_{\text{corr}}(s_t, a_t)$ reward is transformed as $r_{\text{corr}}(s_t, a_t) \rightarrow r_{\text{corr}}(s_t, a_t) + \gamma\Phi(s_{t+1}) - \Phi(s_t)$, where the potential function $\Phi(\cdot)$ can be any function that depends only on the state. Inspired by this structure and prior work [30, 31], we use the following parameterized potential function $\Phi_\theta(s_t)$:

$$\Phi_\theta(x, y_{0:t-1}) = \gamma^{-t} \sum_{i=0}^{t-1} \log \frac{\pi_\theta(y_i|x, y_{0:i-1})}{\pi_{\text{ref}}(y_i|x, y_{0:i-1})},$$

where π_θ is a language model parameterized by θ and π_{ref} is a fixed reference language model. When $i = 0$, the conditional probability $\pi(y_0|x, y_{0:-1})$ reduces to $\pi(y_0|x)$. By construction, this potential function $\Phi(s_t)$ depends only on the state $s_t = (x, y_{0:t-1})$. With the constructed potential function, the corresponding shaping reward function is:

$$r_\theta(s_t, a_t) = r_{\text{corr}}(s_t, a_t) + \gamma\Phi(s_{t+1}) - \Phi(s_t) = r_{\text{corr}}(s_t, a_t) + \gamma^{-t} \log \frac{\pi_\theta(y_t|x, y_{0:t-1})}{\pi_{\text{ref}}(y_t|x, y_{0:t-1})}. \quad (1)$$

4.2 Bilevel optimization

We aim to optimize the shaping reward function r_θ such that the corresponding policy π_{r_θ} can generate correct queries, i.e., maximize the original correctness reward. We formulate this problem as the following bilevel optimization problem:

$$\max_\theta J(\theta) = E_{x\sim\mathcal{D}}^{\pi_{r_\theta}} \Big[\sum_{t=0}^{T} \gamma^t r_{\text{corr}}(S_t, A_t) \Big| S_0 = x \Big] \quad (2)$$

$$\pi_{r_\theta} = \arg\max_\pi E_{x\sim\mathcal{D}}^{\pi} \Big[\sum_{t=0}^{T} \gamma^t \Big(r_\theta(S_t, A_t) - \beta \log \frac{\pi(A_t|S_t)}{\pi_{\text{ref}}(A_t|S_t)} \Big) \Big| S_0 = x \Big], \quad (3)$$

where β is a coefficient and π_{ref} is a fixed reference language model. In the paper, we use capital letters (S_t, A_t) to denote random variables at time t and lower-case letters (s_t, a_t) to denote specific state and action at time t.

The upper-level problem (2) learns a shaping reward function r_θ such that the corresponding policy π_{r_θ} maximizes the original correctness reward function r_{corr}. The lower-level problem (3) learns the corresponding policy π_{r_θ} by solving an entropy-regularized RL problem. The first term in (3) encourages the policy to maximize the learned shaping reward function r_θ and the second term penalizes deviation from the reference policy π_{ref} via a KL-style regularization.

Remark on the distinction between π_θ and π_{r_θ}. We emphasize the distinction between the language model π_θ used to parameterize the shaping reward and the policy π_{r_θ} obtained by solving the lower-level optimization problem (3). The parameter θ defines a language model π_θ, which is used solely to parameterize the shaping reward function r_θ through (1). In contrast, π_{r_θ} denotes the policy that results from optimizing the lower-level RL objective under the shaping reward r_θ. As such, π_{r_θ} is not equal to π_θ; rather, it is implicitly determined by r_θ through the solution of the lower-level problem. Consequently, π_{r_θ} is hyper-parameterized by θ via the shaping reward, even though it has its own policy parameters.

Intuitively, the learned shaping reward acts as an intermediate layer between the original correctness reward and the policy. In the standard setup, the sequence-level correctness reward directly supervises the language model. However, this reward is sparse and provides limited guidance for optimization. To address this issue, we introduce a token-level shaping reward as an intermediate supervision signal. With this design, the language model is optimized using the dense, token-level shaping reward, which provides fine-grained credit assignment and facilitates optimization. At the same time, to ensure that the shaping reward guides the policy toward generating correct queries, the shaping reward itself is supervised by the correctness reward through the upper-level optimization. This hierarchical structure enables dense supervision for policy learning while preserving alignment with the original task objective of generating correct queries.

Algorithm 1 Reward shaping for query generation (RESHAPE)

Input: Reference policy π_{ref}, initial shaping parameter θ_0, prompt set $\mathcal{D}$, and correctness reward function r_{corr}.

Output: Learned shaping parameter θ_K and policy π_K.

1: Initialize policy $\pi_0 = \pi_{\text{ref}}$
2: **for** $k = 0, 1, \cdots, K-1$ **do**
3: Sample a batch of B prompts $\{x^{(n)}\}_{n=1}^{B}$ from $\mathcal{D}$
4: **Policy update**: Optimize the lower-level problem (3) by one-step soft policy iteration to obtain an updated policy π_{k+1}
5: **Reward update**: Update the shaping reward parameter θ via (4).
6: **end for**

5 Algorithm

In this section, we develop an efficient single-loop algorithm to solve the bilevel optimization problem (2)-(3). A straightforward and intuitive way to solve the bilevel optimization problem is double-loop algorithms [32, 33, 36] which consists of an inner loop for solving the lower-level problem and an outer loop for solving the upper-level problem. At each outer iteration, these methods approximately solve the lower-level problem to optimality in the inner loop before performing an update of the upper-level parameters. While double-loop algorithms are theoretically sound, they are computationally prohibitive in our setting, as they require repeatedly solving the lower-level optimization problem in the inner loop at each outer iteration. This computational cost is particularly severe for large language models. To address this challenge, we propose a single-loop algorithm that performs one-step policy update and one-step reward update at each iteration. This design enables efficient joint learning of the shaping reward and the policy while avoiding nested optimization, making the approach computationally efficient at each iteration. In the following context, we elaborate the policy update and reward update.

Policy update. The lower-level optimization problem (3) corresponds to a standard entropy-regularized reinforcement learning objective and can be solved by soft policy iteration [37], which we elaborate in Appendix A.5. In practice, soft policy iteration can be approximated using existing RL algorithms such as GRPO [38], ReMax [11], and RLOO [39]. However, in our single-loop framework, the policy update at iteration k does not require solving the lower-level problem to optimality. Instead, we perform one-step soft policy iteration, resulting in an updated policy π_{k+1}. In practice, this can be approximated by one-step GRPO/RLOO/ReMax gradient ascent of the objective in (3) with respect to the policy. Importantly, Algorithm 1 is agnostic to the choice of RL algorithm. Different off-the-shelf RL algorithms can be used to carry out the one-step policy update, making the framework flexible and compatible with a wide range of existing policy optimization methods.

Reward update. At each iteration k, we update the shaping reward parameter θ by a one-step gradient ascent on the upper-level objective function $J(\theta)$ in (2). This requires computing the hyper-gradient $\nabla J(\theta)$, i.e., the gradient of the upper-level objective function with respect to θ, which accounts for the implicit dependence of the policy π_{r_θ} on the shaping reward parameter θ. Towards this end, at iteration k, for each sampled prompt $x^{(n)}$, we use the latest policy π_{k+1} to generate m responses $\{s_t^{(n),j}, a_t^{(n),j}\}_{j=1}^{m}$ where $s_0^{(n),j} = x^{(n)}$ for all j. These sampled responses are then used to estimate the hyper-gradient. We formalize this estimation in the following lemma.

Lemma 1. *The hyper-gradient can be estimated by*

$$\hat{\nabla} J(\theta_k) = \frac{1}{m\beta B} \sum_{n=1}^{B} \sum_{j=1}^{m} \sum_{t=0}^{T} \gamma^t \Big(\sum_{i=t}^{T} \gamma^i \nabla_\theta r_{\theta_k}(s_i^{(n),j}, a_i^{(n),j}) - \frac{1}{m} \sum_{j'=1}^{m} \sum_{i=t}^{T} \gamma^i \nabla_\theta r_{\theta_k}(s_i^{(n),j'}, a_i^{(n),j'}) \Big) r_{\text{corr}}(s_T^{(n),j}, a_T^{(n),j}),$$

where the correctness reward $r_{\text{corr}}(s_T^{(n),j}, a_T^{(n),j})$ provides a binary feedback at the end of the sequence to indicate whether the corresponding query in the sequence is correct or not. Note that these m sampled responses are also used for the lower-level policy update in the next iteration; therefore, the

reward update does not require additional samples. We use one-step gradient ascent to update the shaping reward parameter

$$\theta_{k+1} = \theta_k + \alpha_k \hat{\nabla} J(\theta_k), \tag{4}$$

where α_k is the step size.

Remark on comparison to PPO. While PPO's critic and RESHAPE's shaping reward may appear structurally similar, they differ fundamentally in the objective, structure, and update rule. We include a detailed discussion and an empirical comparison to elaborate their distinctions in Appendix B.1.1.

6 Theoretical analysis

In this section, we establish the convergence guarantee for RESHAPE (Algorithm 1). A key technical challenge arises from the fact that, at each iteration k, the lower-level policy is updated by only one step. As a result, the updated policy π_{k+1} generally differs from the exact optimizer π_{θ_k} of the lower-level problem (3). This discrepancy introduces a bias in the hyper-gradient estimate, since $\hat{\nabla} J(\theta_k)$ is computed using π_{k+1} rather than π_{θ_k}.

To address this challenge, we adopt the idea of two-timescale stochastic approximation [34, 40] where the lower level updates in a faster timescale and the upper level updates in a slower timescale. The policy update is faster because it converges linearly under a fixed reward function [37] while the reward update is slower given that we choose $\alpha_k \propto (k+1)^{-1/2}$. Intuitively, since the policy update is faster than the reward update, the reward parameter is "relatively fixed" compared to the policy. It is expected that π_{k+1} shall progressively approach π_{θ_k} and at last converge to π_{θ_k} when k increases.

Assumption 1. *The parameterized policy π_θ satisfies $|\log \pi_{\theta_1}(s,a) - \log \pi_{\theta_2}(s,a)| \leq C||\theta_1 - \theta_2||$ and $||\nabla_\theta \log \pi_{\theta_1}(s,a) - \nabla_\theta \log \pi_{\theta_2}(s,a)|| \leq \bar{C}||\theta_1 - \theta_2||$ for any (θ_1, θ_2) and any $(s,a) \in \mathcal{S} \times \mathcal{A}$ where C and $\bar{C}$ are positive constants.*

Assumption 1 is a standard assumption in RL [36, 41, 42, 43, 44, 45, 46, 47] and we justify in Appendix A.3 that this assumption holds when π_θ is a language model. In the following, we quantify the convergence of the lower-level policy update and the convergence of the overall algorithm.

Lemma 2 (lower-level policy convergence)**.** *Suppose Assumption 1 holds, the lower level in Algorithm 1 uses one-step soft policy iteration, and choose $\alpha_k \propto (k+1)^{-1/2}$, then*

$$\frac{1}{K}\sum_{k=0}^{K-1} |\log \pi_{k+1}(a|s) - \log \pi_{r_{\theta_k}}(a|s)| \leq O(\frac{1}{\sqrt{K}}) + O(\frac{1}{K}) \qquad \forall (s,a) \in \mathcal{S} \times \mathcal{A}.$$

Lemma 2 indicates that the lower-level one-step policy update drives π_{k+1} to the corresponding optimal policy π_{θ_k} and will converge to the optimal policy when k increases.

Theorem 1 (upper-level reward convergence)**.** *Suppose the conditions in Lemma 2 hold, then*

$$\frac{1}{K}\sum_{k=0}^{K-1} E\Big[||\nabla J(\theta_k)||^2\Big] \leq O(\frac{1}{\sqrt{K}}) + O(\frac{\log K}{\sqrt{K}}).$$

Theorem 1 indicates that Algorithm 1 converges even if the hyper-gradient has approximation error $||\hat{\nabla} J(\theta) - \nabla J(\theta)||$. This is because that the hyper-gradient approximation error diminishes as π_{k+1} approaches the corresponding optimal policy π_{θ_k} when k increases.

7 Experiments

In this section, we consider two query generation scenarios: SQL for relational databases and Cypher for knowledge graphs. For SQL query generation, we evaluate on two public benchmarks, BIRD [48] and Spider [49]. For Cypher query generation, we conduct experiments on two created knowledge graphs, Finance and Sports. The details of the data collection pipeline for the Cypher benchmarks are included in Appendix B.2. For all the benchmarks, we partition the training sets into two 1:1 subsets where the first subset is SFT set and the second is RL set. RESHAPE (Algorithm 1) and all the RL algorithms are trained after SFT. The experiment details are in Appendix B.

Table 1: Correctness rate (in percentage) for Cypher/SQL query generation across four benchmarks.

Model	Policy Optimizer	Reward	BIRD	Spider	Sports	Finance
Qwen2.5-Coder-3B-Instruct	SFT	N/A	47.25 ± 1.03	79.88 ± 1.18	33.25 ± 0.62	52.63 ± 1.01
	GRPO	Correctness	52.21 ± 0.46	82.11 ± 0.62	36.11 ± 0.87	54.50 ± 0.93
		Composite	52.89 ± 0.87	82.77 ± 1.04	36.82 ± 0.62	55.04 ± 0.82
		PRIME	52.78 ± 0.66	83.04 ± 0.45	37.15 ± 0.73	54.21 ± 0.49
		RESHAPE	$\mathbf{55.49 \pm 0.63}$	$\mathbf{86.02 \pm 0.61}$	$\mathbf{39.24 \pm 0.68}$	$\mathbf{58.85 \pm 0.87}$
	ReMax	Correctness	50.85 ± 0.79	82.17 ± 0.81	35.25 ± 0.93	54.21 ± 0.79
		Composite	52.05 ± 0.78	84.12 ± 0.69	35.87 ± 0.78	54.93 ± 0.53
		PRIME	51.69 ± 0.62	84.01 ± 0.77	35.19 ± 0.64	55.39 ± 0.63
		RESHAPE	$\mathbf{54.32 \pm 0.55}$	$\mathbf{87.64 \pm 0.72}$	$\mathbf{38.02 \pm 0.56}$	$\mathbf{57.32 \pm 0.71}$
	RLOO	Correctness	52.25 ± 1.07	83.16 ± 0.86	35.93 ± 0.81	54.46 ± 1.12
		Composite	53.69 ± 0.94	83.52 ± 0.67	35.26 ± 0.66	$\mathbf{56.81 \pm 0.84}$
		PRIME	54.22 ± 0.46	84.28 ± 0.62	36.09 ± 0.36	55.19 ± 0.64
		RESHAPE	$\mathbf{56.69 \pm 0.73}$	$\mathbf{87.52 \pm 0.51}$	$\mathbf{38.79 \pm 0.51}$	56.77 ± 0.72
deepseek-coder-6.7b-instruct	SFT	N/A	49.65 ± 0.72	80.14 ± 0.96	40.10 ± 0.43	59.79 ± 0.62
	GRPO	Correctness	51.97 ± 0.88	82.46 ± 0.71	40.45 ± 0.59	60.82 ± 0.46
		Composite	53.44 ± 0.55	83.89 ± 0.98	40.38 ± 0.75	60.97 ± 0.57
		PRIME	52.98 ± 1.04	84.22 ± 0.48	40.89 ± 0.76	61.38 ± 0.71
		RESHAPE	$\mathbf{56.50 \pm 0.61}$	$\mathbf{87.11 \pm 0.70}$	$\mathbf{43.45 \pm 0.61}$	$\mathbf{64.01 \pm 0.72}$
	ReMax	Correctness	50.25 ± 0.84	80.84 ± 0.70	40.65 ± 0.28	60.37 ± 1.00
		Composite	50.91 ± 0.74	82.96 ± 0.50	40.83 ± 0.79	61.03 ± 0.42
		PRIME	51.87 ± 0.56	81.66 ± 0.87	41.04 ± 0.75	61.98 ± 0.53
		RESHAPE	$\mathbf{54.40 \pm 0.67}$	$\mathbf{87.04 \pm 0.62}$	$\mathbf{43.97 \pm 0.50}$	$\mathbf{64.22 \pm 0.46}$
	RLOO	Correctness	50.88 ± 0.85	80.46 ± 0.77	40.28 ± 0.65	60.06 ± 0.93
		Composite	51.23 ± 0.61	81.42 ± 1.07	41.04 ± 0.76	60.75 ± 0.61
		PRIME	52.78 ± 0.65	82.26 ± 0.64	42.01 ± 0.75	61.05 ± 0.49
		RESHAPE	$\mathbf{55.14 \pm 0.58}$	$\mathbf{86.02 \pm 0.66}$	$\mathbf{43.32 \pm 0.71}$	$\mathbf{63.74 \pm 0.79}$
Qwen2.5-Coder-7B-Instruct	SFT	N/A	54.24 ± 0.68	84.28 ± 0.76	41.55 ± 0.60	60.42 ± 0.78
	GRPO	Correctness	56.01 ± 0.68	85.85 ± 0.65	42.26 ± 0.49	61.14 ± 0.58
		Composite	57.14 ± 1.12	86.31 ± 0.64	42.44 ± 0.44	61.55 ± 0.65
		PRIME	57.61 ± 0.55	86.24 ± 0.61	42.89 ± 0.56	62.43 ± 0.71
		RESHAPE	$\mathbf{59.91 \pm 0.62}$	$\mathbf{89.04 \pm 0.47}$	$\mathbf{46.32 \pm 0.78}$	$\mathbf{65.47 \pm 0.89}$
	ReMax	Correctness	55.21 ± 0.72	85.62 ± 0.83	41.51 ± 1.21	60.94 ± 0.76
		Composite	55.71 ± 0.66	86.02 ± 0.51	41.88 ± 0.78	60.80 ± 0.62
		PRIME	55.05 ± 0.51	85.52 ± 0.93	42.19 ± 0.74	61.57 ± 0.65
		RESHAPE	$\mathbf{58.52 \pm 0.71}$	$\mathbf{87.68 \pm 0.75}$	$\mathbf{45.96 \pm 0.76}$	$\mathbf{64.82 \pm 0.51}$
	RLOO	Correctness	54.81 ± 1.05	85.03 ± 0.69	42.08 ± 0.69	60.74 ± 0.64
		Composite	55.04 ± 0.62	84.99 ± 0.77	41.85 ± 0.78	61.09 ± 0.86
		PRIME	55.62 ± 1.02	84.39 ± 0.78	43.19 ± 0.89	61.88 ± 0.87
		RESHAPE	$\mathbf{58.05 \pm 0.66}$	$\mathbf{87.98 \pm 0.89}$	$\mathbf{46.08 \pm 0.86}$	$\mathbf{64.66 \pm 0.88}$

Models and baselines. We evaluate RESHAPE using three base language models: Qwen2.5-Coder-3B-Instruct, deepseek-coder-6.7b-instruct, and Qwen2.5-Coder-7B-Instruct. We consider three representative RL algorithms as policy optimizers: GRPO [38], ReMax [11], and RLOO [39]. Algorithm 1 is agnostic to the choice of lower-level policy update algorithm; any of these RL algorithms can be used to perform the one-step policy update. Accordingly, for each RL algorithm, we compare to three baselines: (1) **RL+correctness reward**: this baseline uses only the correctness reward to train an RL algorithm. (2) **RL+composite reward**: this baseline uses the composite reward in SQL-R1 [5] that includes both the correctness reward and other partial rewards such as format reward and length reward. Note that SQL-R1 is originally designed for GRPO, and we adapt it to RLOO and ReMax. While the composite reward design in SQL-R1 is for SQL queries, we adapt it to the Cypher query setting (detailed in Appendix B.2). (3) **RL+PRIME**: this baseline uses PRIME [31] to learn a token-level reward model via cross-entropy loss over the binary correctness labels. PRIME is originally designed for RLOO, and we adapt it to GRPO and ReMax.

Metric and evaluation. Across all benchmarks, we report the correctness rate, defined as the percentage of test prompts for which the model produces a correct answer. For SQL benchmarks, a prediction is considered correct if it exactly matches the ground-truth answer. For the Cypher benchmarks, we determine correctness using a human-validated evaluation pipeline (see Appendix B.2 for details). The evaluation results are calculated from three random seeds.

The results in Table 1 demonstrate that RESHAPE outperforms the baselines across benchmarks and models. Compared to correctness/composite reward that provides sparse supervision, PRIME and RESHAPE achieve better results because they provide dense token-level supervision. In addition,

RESHAPE outperforms PRIME. While PRIME learns token-level rewards via supervised objectives (cross-entropy) to approximate binary correctness labels, our method learns a shaping reward such that the corresponding policy maximizes the correctness reward via RL. Our design directly enforces policy-level alignment: the shaping reward is optimized through its downstream effect on policy performance. In contrast, PRIME indirectly encourages policy-level alignment by approximating correctness signals at the reward level.

RL training from base models. While Table 1 provides results for RL training from the SFT model, we provide additional results of RL training from the base model in Appendix B.1.2. The results in Appendix B.1.2 demonstrate that RESHAPE outperforms the baselines.

Local error corrections. As discussed in Section 3.2, RESHAPE has the potential to reduce local errors by penalizing specific erroneous tokens. To validate this, we conduct a human analysis on a randomly sampled subset of 500 test examples using Qwen2.5-Coder-3B-Instruct+GRPO (Finance), comparing RESHAPE to the best sparse-reward baseline (composite reward). The local errors are categorized into: (1) entity error (misspell entity name); (2) property error (misspell property name); (3) syntax error, and (4) schema-linking error (use wrong class). We also consider joint error, i.e., multiple local errors in a query. Non-local errors (e.g., logical errors) are not included in this analysis.

Table 2: Local error comparison.

	entity error	property error	syntax error	schema-linking error	joint error
Composite reward	4	8	5	5	3
RESHAPE	1	1	0	0	0

The results in Table 2 show that RESHAPE substantially reduces all types of local errors. These errors correspond to localized token-level mistakes, e.g., incorrect entity name or incorrect property name. Moreover, the reduction in local-error rate among the 500 samples is $23/500 = 4.6\%$, which closely matches the overall improvement in Table 1 (3.81%). This suggests that the performance gain is attributed to eliminating local errors. In addition to the quantitative analysis in Table 2, we provide qualitative examples in Appendix B.1 to further illustrate how RESHAPE reduces local errors.

Computation comparison. Although RESHAPE and PRIME introduce additional computation due to reward updates, they reuse the samples generated for policy updates. As a result, the extra overhead is modest, since policy rollout dominates the overall computation cost. We evaluate computation time on the Spider and normalize all results so that the fastest method has a computation time of 1.

Table 3: Computation time comparison.

Model+RL	correctness reward	composite reward	RESHAPE	PRIME
Qwen2.5-Coder-3B-Instruct+GRPO	1.00 ± 0.05	1.06 ± 0.04	1.18 ± 0.08	1.15 ± 0.06
deepseek-coder-6.7b-instruct+ReMax	1.00 ± 0.03	1.04 ± 0.05	1.16 ± 0.05	1.14 ± 0.07
Qwen2.5-Coder-7B-Instruct+RLOO	1.00 ± 0.07	1.02 ± 0.04	1.14 ± 0.03	1.15 ± 0.04

The results in Table 3 show that RESHAPE incurs moderately higher computation time than correctness and composite rewards, and is comparable to PRIME (details in Appendix B.3).

Ablation studies. We conduct comprehensive ablation studies in Appendix C.

8 Conclusion

This paper proposes RESHAPE, an algorithm that learns a token-level reward function to optimize language model query generation. This method is motivated by the key insight that the token-level rewards are particularly helpful for query generation tasks where tiny local errors can invalidate an entire query, in contrast to standard natural language generation tasks. The learned token-level reward can penalize the local errors rather than the entire query, which makes policy optimization more targeted. We formulate a bilevel optimization and propose an efficient single-loop algorithm to solve the problem. Empirical results on SQL query generation and Cypher query generation demonstrate that our method outperforms the baseline methods.

References

[1] E. Bisong, "Google bigquery," in *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners*, pp. 485–517, Springer, 2019.

[2] A. Ferrari and M. Russo, *Introducing Microsoft Power BI*. Microsoft Press, 2016.

[3] R. Nadipalli, *Effective business intelligence with QuickSight*. Packt Publishing Ltd, 2017.

[4] M. Pourreza, S. Talaei, R. Sun, X. Wan, H. Li, A. Mirhoseini, A. Saberi, S. Arik, *et al.*, "Reasoning-SQL: Reinforcement learning with SQL tailored partial rewards for reasoning-enhanced text-to-SQL," in *Conference on Language Modeling*, 2025.

[5] P. Ma, X. Zhuang, C. Xu, X. Jiang, R. Chen, and J. Guo, "SQL-R1: Training natural language to SQL reasoning model by reinforcement learning," *Advances in Neural Information Processing Systems*, 2025.

[6] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, vol. 1. MIT press Cambridge, 1998.

[7] X. Dong, C. Zhang, Y. Ge, Y. Mao, Y. Gao, J. Lin, D. Lou, *et al.*, "C3: Zero-shot text-to-SQL with ChatGPT," *arXiv preprint arXiv:2307.07306*, 2023.

[8] M. Pourreza and D. Rafiei, "Din-SQL: Decomposed in-context learning of text-to-SQL with self-correction," *Advances in Neural Information Processing Systems*, vol. 36, pp. 36339–36348, 2023.

[9] D. Gao, H. Wang, Y. Li, X. Sun, Y. Qian, B. Ding, and J. Zhou, "Text-to-SQL empowered by large language models: A benchmark evaluation," *Proceedings of the VLDB Endowment*, vol. 17, no. 5, pp. 1132–1145, 2024.

[10] Z. Zhang, K. Sun, Z. Tang, J.-S. Vöckler, T. H. Nguyen, and T. Vu, "CypherSmith: Transforming text-to-cypher generation for LLMs with synthetic data," in *Annual Meeting of the Association for Computational Linguistics*, 2026.

[11] H. Li, J. Zhang, H. Liu, J. Fan, X. Zhang, J. Zhu, R. Wei, H. Pan, C. Li, and H. Chen, "Codes: Towards building open-source language models for text-to-SQL," *Proceedings of the ACM on Management of Data*, vol. 2, no. 3, pp. 1–28, 2024.

[12] H. Li, S. Wu, X. Zhang, X. Huang, J. Zhang, F. Jiang, S. Wang, T. Zhang, J. Chen, R. Shi, *et al.*, "OmniSQL: Synthesizing high-quality text-to-SQL data at scale," *arXiv preprint arXiv:2503.02240*, 2025.

[13] M. Pourreza and D. Rafiei, "DTS-SQL: Decomposed text-to-SQL with small large language models," in *Empirical Methods on Natural Language Processing*, pp. 8212–8220, 2024.

[14] Z. Zhang, T. Vu, and A. Moschitti, "Joint models for answer verification in question answering systems," in *Annual Meeting of the Association for Computational Linguistics*, pp. 3252–3262, 2021.

[15] Z. Zhang, T. Vu, S. Gandhi, A. Chadha, and A. Moschitti, "Wdrass: A web-scale dataset for document retrieval and answer sentence selection," in *ACM International Conference on Information & Knowledge Management*, p. 4707–4711, 2022.

[16] Z. Zhang, T. Vu, and A. Moschitti, "Double retrieval and ranking for accurate question answering," in *Findings of the Association for Computational Linguistics*, 2023.

[17] Z. Zhang, T. Vu, and A. Moschitti, "In situ answer sentence selection at web-scale," in *ACM International Conference on Information and Knowledge Management*, p. 4298–4302, 2024.

[18] M. Nguyen, T. Q. Nguyen, K. KC, Z. Zhang, and T. Vu, "Reinforcement Learning from Answer Reranking Feedback for Retrieval-Augmented Answer Generation," in *Interspeech 2024*, pp. 4044–4048, 2024.

[19] Z. Zhang, A. Moschitti, and T. Vu, "Retrieving support to rank answers in open-domain question answering," in *Conference on Empirical Methods in Natural Language Processing*, 2025.

[20] V. Zhong, C. Xiong, and R. Socher, "Seq2SQL: Generating structured queries from natural language using reinforcement learning," *arXiv preprint arXiv:1709.00103*, 2017.

[21] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *International Conference on Machine Learning*, vol. 99, pp. 278–287, 1999.

[22] E. Wiewiora, "Potential-based shaping and q-value initialization are equivalent," *Journal of Artificial Intelligence Research*, vol. 19, pp. 205–208, 2003.

[23] M. J. Mataric, "Reward functions for accelerated learning," in *International Conference on Machine Learning*, pp. 181–189, 1994.

[24] A. Laud and G. DeJong, "The influence of reward on the speed of reinforcement learning: An analysis of shaping," in *International Conference on Machine Learning*, pp. 440–447, 2003.

[25] S. Singh, R. L. Lewis, and A. G. Barto, "Where do rewards come from," in *Annual Conference of the Cognitive Science Society*, pp. 2601–2606, 2009.

[26] G. Konidaris and A. Barto, "Skill discovery in continuous reinforcement learning domains using skill chaining," *Advances in Neural Information Processing Systems*, vol. 22, 2009.

[27] Z. Zheng, J. Oh, and S. Singh, "On learning intrinsic rewards for policy gradient methods," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[28] Y. Hu, W. Wang, H. Jia, Y. Wang, Y. Chen, J. Hao, F. Wu, and C. Fan, "Learning to utilize shaping rewards: A new approach of reward shaping," *Advances in Neural Information Processing Systems*, vol. 33, pp. 15931–15941, 2020.

[29] S. Liu and M. Zhu, "UTILITY: Utilizing explainable reinforcement learning to improve reinforcement learning," in *International Conference on Learning Representations*, 2025.

[30] L. Yuan, W. Li, H. Chen, G. Cui, N. Ding, K. Zhang, B. Zhou, Z. Liu, and H. Peng, "Free process rewards without process labels," in *International Conference on Machine Learning*, 2025.

[31] G. Cui, L. Yuan, Z. Wang, H. Wang, Y. Zhang, J. Chen, W. Li, B. He, Y. Fan, T. Yu, *et al.*, "Process reinforcement through implicit rewards," *arXiv preprint arXiv:2502.01456*, 2025.

[32] S. Ghadimi and M. Wang, "Approximation methods for bilevel programming," *arXiv preprint arXiv:1802.02246*, 2018.

[33] K. Ji, J. Yang, and Y. Liang, "Bilevel optimization: Convergence analysis and enhanced design," in *International Conference on Machine Learning*, pp. 4882–4892, 2021.

[34] M. Hong, H.-T. Wai, Z. Wang, and Z. Yang, "A two-timescale stochastic algorithm framework for bilevel optimization: Complexity analysis and application to actor-critic," *SIAM Journal on Optimization*, vol. 33, no. 1, pp. 147–180, 2023.

[35] T. Chen, Y. Sun, Q. Xiao, and W. Yin, "A single-timescale method for stochastic bilevel optimization," in *International Conference on Artificial Intelligence and Statistics*, pp. 2466–2488, 2022.

[36] S. Liu and M. Zhu, "Distributed inverse constrained reinforcement learning for multi-agent systems," *Advances in Neural Information Processing Systems*, vol. 35, pp. 33444–33456, 2022.

[37] S. Cen, C. Cheng, Y. Chen, Y. Wei, and Y. Chi, "Fast global convergence of natural policy gradient methods with entropy regularization," *Operations Research*, vol. 70, no. 4, pp. 2563–2578, 2022.

[38] Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y. Li, Y. Wu, *et al.*, "Deepseekmath: Pushing the limits of mathematical reasoning in open language models," *arXiv preprint arXiv:2402.03300*, 2024.

[39] A. Ahmadian, C. Cremer, M. Gallé, M. Fadaee, J. Kreutzer, O. Pietquin, A. Üstün, and S. Hooker, "Back to Basics: Revisiting reinforce-style optimization for learning from human feedback in llms," in *Annual Meeting of the Association for Computational Linguistics*, pp. 12248–12267, 2024.

[40] S. Zeng, C. Li, A. Garcia, and M. Hong, "Maximum-likelihood inverse reinforcement learning with finite-time guarantees," *Advances in Neural Information Processing Systems*, vol. 35, pp. 10122–10135, 2022.

[41] H. Kumar, A. Koppel, and A. Ribeiro, "On the sample complexity of actor-critic method for reinforcement learning with function approximation," *Machine Learning*, vol. 112, no. 7, pp. 2433–2467, 2023.

[42] S. Liu and M. Zhu, "Learning multi-agent behaviors from distributed and streaming demonstrations," *Advances in Neural Information Processing Systems*, vol. 36, pp. 53552–53564, 2023.

[43] K. Zhang, A. Koppel, H. Zhu, and T. Basar, "Global convergence of policy gradient methods to (almost) locally optimal policies," *SIAM Journal on Control and Optimization*, vol. 58, no. 6, pp. 3586–3612, 2020.

[44] S. Liu and M. Zhu, "Meta inverse constrained reinforcement learning: Convergence guarantee and generalization analysis," in *International Conference on Learning Representations*, 2024.

[45] S. Liu and M. Zhu, "In-trajectory inverse reinforcement learning: Learn incrementally before an ongoing trajectory terminates," *Advances in Neural Information Processing Systems*, vol. 37, pp. 117164–117209, 2024.

[46] S. Liu, S. Xu, W. Qiu, H. Zhang, and M. Zhu, "Explainable reinforcement learning from human feedback to improve alignment," *Advances in Neural Information Processing Systems*, vol. 38, pp. 138261–138291, 2025.

[47] S. Liu and M. Zhu, "Leveraging explanation to improve generalization of meta reinforcement learning," in *International Conference on Learning Representations*, 2026.

[48] J. Li, B. Hui, G. Qu, J. Yang, B. Li, B. Li, B. Wang, B. Qin, R. Geng, N. Huo, *et al.*, "Can LLM already serve as a database interface? A big bench for large-scale database grounded text-to-SQLs," *Advances in Neural Information Processing Systems*, vol. 36, pp. 42330–42357, 2023.

[49] T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman, *et al.*, "Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task," in *Empirical Methods in Natural Language Processing*, pp. 3911–3921, 2018.

[50] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, "Reinforcement learning with deep energy-based policies," in *International Conference on Machine Learning*, pp. 1352–1361, 2017.

[51] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International Conference on Machine Learning*, pp. 1861–1870, 2018.

[52] Y. Nesterov, *Introductory lectures on convex optimization: A basic course*, vol. 87. Springer Science & Business Media, 2013.

[53] S. Bubeck *et al.*, "Convex optimization: Algorithms and complexity," *Foundations and Trends® in Machine Learning*, vol. 8, no. 3-4, pp. 231–357, 2015.

This appendix includes two sections: proof and experiment details.

A Proof

This section includes the proof of all the theoretical statements in the paper.

A.1 Preliminary theoretical results

In this subsection, we provide well-established theoretical results from the literature that serve as the building blocks of our theoretical analysis.

Lemma 3 ([50, 51])**.** *The optimal policy π_{r_θ} of the lower-level optimization problem (3) is the soft policy:*

$$\begin{aligned}
\pi_{r_\theta,t}(a_t|s_t) &= \frac{\pi_{\text{ref},t}(a_t|s_t)\exp(\frac{1}{\beta}Q^*_{r_\theta,t}(s_t,a_t))}{\sum_{a'\in\mathcal{A}}\pi_{\text{ref},t}(a'|s_t)\exp(\frac{1}{\beta}Q^*_{r_\theta,t}(s_t,a'))}\\
Q^*_{r_\theta,t}(s_t,a_t) &= r_\theta(s_t,a_t)+\gamma V^*_{r_\theta,t+1}(s_{t+1})\\
V^*_{r_\theta,t}(s_t) &= \beta\log\Big[\sum_{a'\in\mathcal{A}}\pi_{\text{ref},t}(a'|s_t)\exp(\frac{1}{\beta}Q^*_{r_\theta,t}(s_t,a'))\Big].
\end{aligned}$$

Note that in standard RL [6], the policy π_{r_θ}, state-value function $V^*_{r_\theta}$, and action-value function $Q^*_{r_\theta}$ should depend on t for finite-horizon MDPs. In the main body, we omit this dependency of t for simple notations and alignment with language model RL literature. In the proof, we include this dependency of t for rigorous analysis. Importantly, including this dependency of t is more general and can reduce to the case where the policy, state-value function and action-value function do not depend on t. Therefore, in the analysis, $\pi_{r_\theta} = (\pi_{r_\theta,0},\cdots,\pi_{r_\theta,T})$. This is also true for π_{ref}, state-value function, and action-value function.

A.2 Proof of Lemma 1

Claim 1. *We have the following gradient:* $\frac{1}{\beta}\nabla_\theta\log\pi_{r_\theta,t}(a_t|s_t) = E^{\pi_{r_\theta}}[\sum_{i=t}^T\gamma^i\nabla_\theta r_\theta(S_i,A_i)|S_t=s_t,A_t=a_t] - \frac{1}{\beta}E^{\pi_{r_\theta}}[\sum_{i=t}^T\gamma^i\nabla_\theta r_\theta(S_i,A_i)|S_t=s_t]$.

Proof. Define $Z_{r_\theta,t}(s_t,a_t)\triangleq\pi_{\text{ref},t}(a_t|s_t)\exp(\frac{1}{\beta}Q^*_{r_\theta,t}(s_t,a_t))$ and $Z_{r_\theta,t}(s_t)\triangleq\exp(\frac{1}{\beta}V^*_{r_\theta,t}(s_t))$

$$\begin{aligned}
&\nabla_\theta\log Z_{r_\theta,t}(s_t) = \frac{\sum_{a_t\in\mathcal{A}}\nabla_\theta Z_{r_\theta,t}(s_t,a_t)}{Z_{r_\theta,,t}(s_t)},\\
&=\sum_{a_t\in\mathcal{A}}\frac{Z_{r_\theta,t}(s_t,a_t)}{Z_{r_\theta,t}(s_t)}\nabla_\theta\log Z_{r_\theta,t}(s_t,a_t),\\
&=\sum_{a_t\in\mathcal{A}}\pi_{r_\theta,t}(a_t|s_t)\Big[\frac{\nabla_\theta r_\theta(s_t,a_t)}{\beta}+\gamma\nabla_\theta\log Z_{r_\theta,t+1}(s_{t+1})\Big],\\
&=\sum_{a_t\in\mathcal{A}}\pi_{\theta,t}(a_t|s_t)\Big[\frac{\nabla_\theta r_\theta(s_t,a_t)}{\beta}\\
&+\gamma\sum_{a_{t+1}\in\mathcal{A}}\pi_{r_\theta,t+1}(a_{t+1}|s_{t+1})\Big(\frac{\nabla_\theta r_\theta(s_{t+1},a_{t+1})}{\beta}+\gamma\nabla_\theta\log Z_{r_\theta,t+2}(s_{t+2})\Big)\Big].
\end{aligned}$$

Keep the expansion, we can get $\nabla_\theta\log Z_{r_\theta,t}(s_t) = \frac{1}{\beta}E^{\pi_{r_\theta}}[\sum_{i=t}^T\gamma^i\nabla_\theta r_\theta(S_i,A_i)|S_t=s_t]$ and similarly we can get $\nabla_\theta\log Z_{r_\theta,t}(s_t,a_t) = \frac{1}{\beta}E^{\pi_{r_\theta}}[\sum_{i=t}^T\gamma^i\nabla_\theta r_\theta(S_i,A_i)|S_t=s_t,A_t=a_t]$. Thus we have the gradient $\nabla_\theta\log\pi_{r_\theta,t}(a_t|s_t) = \nabla_\theta\log Z_{r_\theta,t}(s_t,a_t) - \nabla_\theta\log Z_{r_\theta,t}(s_t)$. □

Now, we can proceed to obtain the hyper-gradient. Recall from (2) that $J(\theta) = E_{x\sim\mathcal{D}}^{\pi_{r_\theta}}\Big[\sum_{t=0}^{T}\gamma^t r_{\text{corr}}(s_t,a_t)\Big]$.

$$\begin{aligned}\nabla J(\theta) &\overset{(a)}{=} E_{x\sim\mathcal{D}}E^{\pi_{r_\theta}}\Big[\sum_{t=0}^{T}\gamma^t\nabla_\theta\log\pi_{r_\theta,t}(A_t|S_t)Q^{\pi_{r_\theta}}_{r_{\text{corr}},t}(S_t,A_t)|S_0=x\Big]\\ &\overset{(b)}{=} E_{x\sim\mathcal{D}}E^{\pi_{r_\theta}}\Big[\sum_{t=0}^{T}\gamma^t\nabla_\theta\log\pi_{r_\theta,t}(A_t|S_t)r_{\text{corr}}(S_T,A_T)|S_0=x\Big]\\ &\overset{(c)}{=} E_{x\sim\mathcal{D}}E^{\pi_{r_\theta}}\Big[\sum_{t=0}^{T}\gamma^t\nabla_\theta\Big(\log Z_{r_\theta,t}(S_t,A_t)-\log Z_{r_\theta,t}(S_t)\Big)r_{\text{corr}}(S_T,A_T)|S_0=x\Big],\end{aligned}\tag{5}$$

where (a) follows the standard policy gradient theorem [6] and the action-value function under π_{r_θ} and r_{corr} is $Q^{\pi_{r_\theta}}_{r_{\text{corr}}}(s_t,a_t)=E^{\pi_{r_\theta}}[\sum_{i=t}^{T}\gamma^i r_{\text{corr}}(S_i,A_i)|S_t=s_t,A_t=a_t]$, (b) follows the fact that r_{corr} is a sequence-level reward function and only assigns the reward value at the terminal state-action pair (S_T,A_T), and (c) follows Claim 1.

Recall from Claim 1 that

$$\begin{aligned}\nabla_\theta\log Z_{r_\theta,t}(s_t,a_t)&=\frac{1}{\beta}E^{\pi_{r_\theta}}[\sum_{i=t}^{T}\gamma^i\nabla_\theta r_\theta(S_i,A_i)|S_t=s_t,A_t=a_t]\\ \nabla_\theta\log Z_{r_\theta,t}(s_t)&=\frac{1}{\beta}E^{\pi_{r_\theta}}[\sum_{i=t}^{T}\gamma^i\nabla_\theta r_\theta(S_i,A_i)|S_t=s_t].\end{aligned}$$

To precisely compute the hyper-gradient (5), at each state s_t, we need to sample multiple trajectories, e.g., sample m trajectories $\{s_t,a_t^j,s_{t+1}^j,\cdots,s_T^j,a_T^j\}_{j=1}^m$. Then we can estimate

$$\begin{aligned}\nabla_\theta\log Z_{r_\theta,t}(s_t,a_t^j)&\approx\frac{1}{\beta}\Big[\gamma^t\nabla_\theta r_\theta(s_t,a_t^j)+\sum_{i=t+1}^{T}\gamma^i\nabla_\theta r_\theta(s_i^j,a_i^j)\Big]\\ \nabla_\theta\log Z_{r_\theta,t}(s_t)&\approx\frac{1}{\beta}\Big[\frac{1}{m}\gamma^t\sum_{j=1}^{m}\nabla_\theta r_\theta(s_t,a_t^j)+\sum_{i=t+1}^{T}\gamma^i\nabla_\theta r_\theta(s_i^j,a_i^j)\Big].\end{aligned}$$

However, it is intractable to estimate $\nabla_\theta\log Z_{r_\theta}(s_t,a_t^j)$ and $\nabla_\theta\log Z_{r_\theta}(s_t)$ at each time t in this way because it requires to sample m trajectories at each time t. To address this issue, we use an approximation method proposed in subsection 4.1.3 of GRPO [38] for token-level reward. Specifically, we only sample m trajectories at the initial state $s_0^{(n)}=x^{(n)}$: $\{s_0^{(n)},a_0^{(n),j},s_1^{(n),j},a_1^{(n),j},\cdots,s_T^{(n),j},a_T^{(n),j}\}_{j=1}^m$. We approximate $\nabla_\theta\log Z_{r_\theta}(S_t)$ via

$$\nabla_\theta\log Z_{r_\theta,t}(S_t)\approx\frac{1}{\beta m}\sum_{j'=1}^{m}\sum_{i=t}^{T}\gamma^i\nabla_\theta r_\theta(s_i^{(n),j'},a_i^{(n),j'}),$$

therefore, we can approximate the hyper-gradient (5) via

$$\begin{aligned}\nabla J(\theta)\approx E_{x^{(n)}\sim\mathcal{D}}\Big[\frac{1}{m}\sum_{j=1}^{m}\sum_{t=0}^{T}\gamma^t\Big(\frac{1}{\beta}\sum_{i=t}^{T}\gamma^i\nabla_\theta r_\theta(s_i^{(n),j},a_i^{(n),j})\\ -\frac{1}{\beta m}\sum_{j'=1}^{m}\sum_{i=t}^{T}\gamma^i\nabla_\theta r_\theta(s_i^{(n),j'},a_i^{(n),j'})\Big)r_{\text{corr}}(s_T^{(n),j},a_T^{(n),j})\Big].\end{aligned}\tag{6}$$

Since we sample B prompts $\{x^{(n)}\}_{n=1}^B$, we can estimate the expectation in (6) via

$$\nabla J(\theta)\approx\frac{1}{m\beta B}\sum_{n=1}^{B}\sum_{j=1}^{m}\sum_{t=0}^{T}\gamma^t\Big(\sum_{i=t}^{T}\gamma^i\nabla_\theta r_\theta(s_i^{(n),j},a_i^{(n),j})$$

$$- \frac{1}{m}\sum_{j=1}^{m}\sum_{i=t}^{T}\gamma^i \nabla_\theta r_\theta(s_i^{(n),j}, a_i^{(n),j})\Big) r_{\text{corr}}(s_T^{(n),j}, a_T^{(n),j}).$$

Note that the m trajectories should be sampled by π_{θ_k} at each iteration k. However, we use a single-loop algorithm and we only have an approximation π_{k+1} of π_{θ_k} at each iteration k. Therefore, we use the policy π_{k+1} to sample the m trajectories. This will induce an approximation error, which we will quantify later.

A.3 Justification of Assumption 1

A transformer-based language model can be generally expressed as $\pi_\theta(a|s) = \frac{\exp(f_\theta(s)_a)}{\sum_{a'}\exp(f_\theta(s)_{s'})}$ where $f_\theta(s)$ is the logit function produced by a transformer. A transformer is a composition of linear layers, attention (bilinear+softmax), layer norm, and activation functions. The aforementioned components are all smooth and we can choose smooth activation functions such as tanh. Therefore, the language model π_θ is smooth in θ and thus $\log \pi_\theta$ is also smooth in θ. In Algorithm 1, we use finite number of iterations K, therefore, the optimization trajectory of θ is bounded within a compact set. Since $\log \pi_\theta$ is smooth in θ and the optimization trajectory of θ is bounded in Algorithm 1, Assumption 1 holds in our case.

A.4 Intermediate theoretical results

In this subsection, we establish some intermediate theoretical results that will be used to prove the convergence of Algorithm 1.

Lemma 4. *The shaping reward function r_θ satisfies $|r_{\theta_1}(s,a) - r_{\theta_2}(s,a)| \le C||\theta_1 - \theta_2||$ and $||\nabla_\theta r_{\theta_1}(s,a) - \nabla_\theta r_{\theta_2}(s,a)|| \le \bar{C}||\theta_1 - \theta_2||$ for any (θ_1, θ_2) and any $(s,a) \in \mathcal{S}\times\mathcal{A}$ where C and $\bar{C}$ are positive constants.*

Proof. This is a direct result based on Assumption 1 because from the parameterization (1), we can see that $\nabla_\theta r_\theta(s,a) = \nabla_\theta \log \pi_\theta(a|s)$. On a convex domain, $|f(x_1) - f(x_2)| \le L||x_1 - x_2||$ and $||\nabla f(x)|| \le L$ are equivalent [52, 53], where L is a positive constant. □

Lemma 5. *It holds that $||\hat{\nabla} J(\theta)|| \le \frac{2CT^2}{\beta}$ and $||\nabla J(\theta)|| \le \frac{2CT^2}{\beta}$ for any θ.*

Proof. Recall from Lemma 1 that

$$\hat{\nabla} J(\theta) = E_{x^{(n)}\sim\mathcal{D}}\Big[\frac{1}{m}\sum_{j=1}^{m}\sum_{t=0}^{T}\gamma^t\Big(\frac{1}{\beta}\sum_{i=t}^{T}\gamma^i\nabla_\theta r_\theta(s_i^{(n),j}, a_i^{(n),j}) - \frac{1}{\beta m}\sum_{j'=1}^{m}\sum_{i=t}^{T}\gamma^i\nabla_\theta r_\theta(s_i^{(n),j'}, a_i^{(n),j'})\Big) r_{\text{corr}}(s_T^{(n),j}, a_T^{(n),j})\Big].$$

Since $||\nabla r_\theta(s,a)|| \le C$ (Lemma 4) and $r_{\text{corr}}(s_T, a_T) \in \{0,1\}$, we have that $||\hat{\nabla} J(\theta)|| \le \frac{2CT^2}{\beta}$. Similarly, we can get $||\nabla J(\theta)|| \le \frac{2CT^2}{\beta}$. □

Lemma 6. *Suppose Assumption 1 holds, then*

$$||\nabla J(\theta_1) - \nabla J(\theta_2)|| \le C_J||\theta_1 - \theta_2||,$$

for any θ_1 and θ_2, where $C_J = \frac{2\bar{C}T^2 + 4C^2T^4}{\beta} + \frac{C^2T^2}{\beta^2}$ is a positive constant.

Proof. Recall from (5) that

$$\nabla J(\theta) = E_{x\sim\mathcal{D}}E^{\pi_{r_\theta}}\Big[\sum_{t=0}^{T}\gamma^t\nabla_\theta\Big(\log Z_{r_\theta,t}(S_t, A_t) - \log Z_{r_\theta,t}(S_t)\Big) r_{\text{corr}}(S_T, A_T)|S_0 = x\Big],$$

where the terms $\nabla_\theta \log Z_{r_\theta,t}(s_t,a_t) = \frac{1}{\beta}E^{\pi_{r_\theta}}[\sum_{i=t}^{T}\gamma^i\nabla_\theta r_\theta(S_i,A_i)|S_t = s_t, A_t = a_t]$ and $\nabla_\theta \log Z_{r_\theta,t}(s_t) = \frac{1}{\beta}E^{\pi_{r_\theta}}[\sum_{i=t}^{T}\gamma^i\nabla_\theta r_\theta(S_i,A_i)|S_t = s_t]$. To find the smoothness constant of $J(\theta)$, we need to compute its Hessian. First, we have that

$$\nabla^2_{\theta\theta}\log Z_{r_\theta,t}(s,a) = \frac{1}{\beta}\nabla_\theta E^{\pi_{r_\theta}}\Big[\sum_{i=t}^{T}\gamma^i\nabla_\theta r_\theta(S_i,A_i)|S_t = s, A_t = a\Big]$$
$$= \frac{\gamma^t}{\beta}\nabla^2_{\theta\theta}r_\theta(s,a) + \frac{1}{\beta}\nabla_\theta E^{\pi_{r_\theta}}\Big[\sum_{i=t+1}^{T}\gamma^i\nabla_\theta r_\theta(S_i,A_i)|S_{t+1} = s'\Big]$$
$$= \frac{\gamma^t}{\beta}\nabla^2_{\theta\theta}r_\theta(s,a) + \frac{1}{\beta}\nabla_\theta \sum_{a'\in\mathcal{A}}\pi_{r_\theta,t+1}(a'|s')E^{\pi_{r_\theta}}\Big[\sum_{i=t+1}^{T}\gamma^i\nabla_\theta r_\theta(S_i,A_i)|S_{t+1} = s', A_{t+1} = a'\Big]$$
$$= \frac{\gamma^t}{\beta}\nabla^2_{\theta\theta}r_\theta(s,a) + \frac{1}{\beta}\sum_{a'\in\mathcal{A}}\nabla_\theta\pi_{r_\theta,t+1}(a'|s')\cdot E^{\pi_{r_\theta}}\Big[\sum_{i=t+1}^{T}\gamma^i\nabla_\theta r_\theta(S_i,A_i)|S_{t+1} = s', A_{t+1} = a'\Big]$$
$$+ \frac{1}{\beta}\sum_{a'\in\mathcal{A}}\pi_{r_\theta,t+1}(a'|s')\cdot\nabla_\theta E^{\pi_{r_\theta}}\Big[\sum_{i=t+1}^{T}\gamma^i\nabla_\theta r_\theta(S_i,A_i)|S_{t+1} = s', A_{t+1} = a'\Big].$$

Keep the expansion, we can get

$$\nabla^2_{\theta\theta}\log Z_{r_\theta,t}(s,a) = \frac{1}{\beta}E^{\pi_{r_\theta}}\Big[\sum_{i=t}^{T}\gamma^i\nabla^2_{\theta\theta}r_\theta(S_i,A_i)|S_t = s, A_t = a\Big]$$
$$+ \frac{1}{\beta}E^{\pi_{r_\theta}}\Big[\sum_{i=t}^{T}\gamma^i\nabla_\theta\pi_{r_\theta}(A_i|S_i)\cdot E^{\pi_{r_\theta}}\Big[\sum_{j=i}^{T}\gamma^j\nabla_\theta r_\theta(S_j,A_j)\Big]\Big|S_t = s, A_t = a\Big].$$

Now we take a look at the second term in the above equation:

$$\nabla_\theta\pi_{r_\theta}(A_i|S_i)\cdot E^{\pi_{r_\theta}}\Big[\sum_{j=i}^{T}\gamma^j\nabla_\theta r_\theta(S_j,A_j)\Big]$$
$$= \pi_{r_\theta}(A_i|S_i)\nabla_\theta\log\pi_{r_\theta}(A_i|S_i)\cdot E^{\pi_{r_\theta}}\Big[\sum_{j=i}^{T}\gamma^j\nabla_\theta r_\theta(S_j,A_j)\Big]$$
$$\overset{(a)}{=} \pi_{r_\theta}(A_i|S_i)\Big[\nabla_\theta\log Z_{r_\theta,t}(S_i,A_i) - \nabla_\theta\log Z_{r_\theta,t}(S_i)\Big]E^{\pi_{r_\theta}}\Big[\sum_{j=i}^{T}\gamma^j\nabla_\theta r_\theta(S_j,A_j)\Big]$$
$$\Rightarrow \Big|\Big|\nabla_\theta\pi_{r_\theta}(A_i|S_i)\cdot E^{\pi_{r_\theta}}\Big[\sum_{j=i}^{T}\gamma^j\nabla_\theta r_\theta(S_j,A_j)\Big]\Big|\Big| \le 2CT\cdot CT = 2C^2T^2. \tag{7}$$

where (a) follows Claim 1. Therefore, we have that

$$||\nabla^2_{\theta\theta}\log Z_{r_\theta,t}(s,a)|| \le \frac{1}{\beta}\Big|\Big|E^{\pi_{r_\theta}}\Big[\sum_{i=t}^{T}\gamma^i\nabla^2_{\theta\theta}r_\theta(S_i,A_i)|S_t = s, A_t = a\Big]\Big|\Big|$$
$$+ \frac{1}{\beta}\Big|\Big|E^{\pi_{r_\theta}}\Big[\sum_{i=t}^{T}\gamma^i\nabla_\theta\pi_{r_\theta}(A_i|S_i)\cdot E^{\pi_{r_\theta}}\Big[\sum_{j=i}^{T}\gamma^j\nabla_\theta r_\theta(S_j,A_j)\Big]\Big|S_t = s, A_t = a\Big]\Big|\Big|$$
$$\overset{(b)}{\le} \frac{\bar{C}T}{\beta} + \frac{1}{\beta}\cdot T\cdot 2C^2T^2 = \frac{\bar{C}T + 2C^2T^3}{\beta}.$$

where (b) follows Lemma 4 and (7). Similarly, we can get $||\nabla^2_{\theta\theta}\log Z_{r_\theta,t}(s)|| \le \frac{\bar{C}T+2C^2T^3}{\beta}$. Therefore, we have that

$$||\nabla^2 J(\theta)|| \le \Big|\Big|E_{x\sim\mathcal{D}}E^{\pi_{r_\theta}}\Big[\sum_{t=0}^{T}\gamma^t\nabla^2_{\theta\theta}\Big(\log Z_{r_\theta,t}(S_t,A_t) - \log Z_{r_\theta,t}(S_t)\Big)r_{\text{corr}}(S_T,A_T)|S_0 = x\Big]\Big|\Big|$$

$$+ \Big|\Big| E_{x\sim\mathcal{D}} E^{\pi_{r_\theta}} \Big[\sum_{t=0}^{T} \gamma^t \nabla_\theta \log \pi_{r_\theta}(A_t|S_t) \cdot \nabla_\theta \Big(\log Z_{r_\theta,t}(S_t, A_t) - \log Z_{r_\theta,t}(S_t) \Big) \cdot$$
$$r_{\text{corr}}(S_T, A_T)|S_0 = x \Big] \Big|\Big|$$
$$\leq \frac{2\bar{C}T^2 + 4C^2T^4}{\beta} + \Big(\frac{CT}{\beta}\Big)^2.$$

□

A.5 Proof of Lemma 2

We use soft policy iteration as the lower-level policy update for analysis. At each iteration k, we have a policy π_k and a reward function r_{θ_k}, a one-step policy update to get π_{k+1} as follows:

$$\pi_{k+1,t}(a|s) = \frac{\pi_{\text{ref},t}(a|s) \exp(\frac{1}{\beta} Q^{\pi_k}_{r_{\theta_k},t}(s,a))}{\sum_{\bar{a}\in\mathcal{A}} \pi_{\text{ref},t}(\bar{a}|s) \exp(\frac{1}{\beta} Q^{\pi_k}_{r_{\theta_k},t}(s,\bar{a}))}$$
$$Q^{\pi_k}_{r_{\theta_k},t}(s,a) = r_{\theta_k}(s,a) + \gamma V^{\pi_k}_{r_{\theta_k},t+1}(s')$$
$$V^{\pi_k}_{r_{\theta_k},t}(s) = \beta \log \Big[\sum_{\bar{a}\in\mathcal{A}} \pi_{\text{ref},t}(\bar{a}|s) \exp(\frac{1}{\beta} Q^{\pi_k}_{r_{\theta_k},t}(s,\bar{a})) \Big],$$

where $s' \sim P(\cdot|s,a)$ (i.e., $s' = (s,a)$) and the terminal value $V^{\pi_k}_{r_{\theta_k},T+1} \equiv 0$ for all k.

Define the time-indexed soft Bellman operator

$$(\mathcal{T}_{\theta_k,t} Q)(s,a) \triangleq r_{\theta_k}(s,a) + \gamma\beta \log \Big[\sum_{\bar{a}'\in\mathcal{A}} \pi_{\text{ref},t}(\bar{a}'|s') \exp(\frac{1}{\beta} Q(s',\bar{a}')) \Big],$$

and define the finite-horizon operator $\mathcal{F}_{\theta_k}$ acting on the action-value functions $Q = (Q_0, \cdots, Q_{T-1})$ by

$$(\mathcal{F}_{\theta_k}(Q))_t = \mathcal{T}_{\theta_k,t}(Q_{t+1}), \qquad t = 0, \ldots, T.$$

Then one-step soft policy iteration is equivalently written as

$$Q^{k+1} = \mathcal{F}_{\theta_k}(Q^k), \qquad \pi_{k,t}(s,a) \propto \pi_{\text{ref},t}(a|s) \exp(\frac{1}{\beta} Q^k_t(s,a)).$$

Claim 2. *For two action-value functions $Q = (Q_0, \cdots, Q_{T-1})$ and $Q' = (Q'_0, \cdots, Q'_{T-1})$ where $Q, Q' \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|\times T}$, it holds that*

$$||\mathcal{F}_{\theta_k}(Q) - \mathcal{F}_{\theta_k}(Q')||_\infty \leq \gamma ||Q - Q'||_\infty.$$

Proof. We define $||Q - Q'||_\infty \triangleq \max_{s\in\mathcal{S}, a\in\mathcal{A}, 0\leq t\leq T} |Q_t(s,a) - Q'_t(s,a)|$ and denote $\epsilon = ||Q - Q'||_\infty$. Therefore, we have that

$$\gamma\beta \log \Big[\sum_{a\in\mathcal{A}} \pi_{\text{ref},t}(a|s) \exp(\frac{1}{\beta} Q_{t+1}(s,a)) \Big] \leq \gamma\beta \log \Big[\sum_{a\in\mathcal{A}} \pi_{\text{ref},t}(a|s) \exp(\frac{1}{\beta}(Q'_{t+1}(s,a) + \epsilon)) \Big]$$
$$= \gamma\epsilon + \gamma\beta \log \Big[\sum_{a\in\mathcal{A}} \pi_{\text{ref},t}(a|s) \exp(\frac{1}{\beta} Q'_{t+1}(s,a)) \Big].$$

Similarly, we can get that

$$\gamma\beta \log \Big[\sum_{a\in\mathcal{A}} \pi_{\text{ref},t}(a|s) \exp(\frac{1}{\beta} Q_{t+1}(s,a)) \Big] \geq -\gamma\epsilon + \gamma\beta \log \Big[\sum_{a\in\mathcal{A}} \pi_{\text{ref},t}(a|s) \exp(\frac{1}{\beta} Q'_{t+1}(s,a)) \Big].$$

Therefore, $||\mathcal{F}_{\theta_k}(Q) - \mathcal{F}_{\theta_k}(Q')||_\infty \leq \gamma\epsilon = \gamma||Q - Q'||_\infty$. □

Claim 3. *It holds that $Q^{\pi_{k+1}}_{r_{\theta_k},t}(s,a) \geq \mathcal{T}_{\theta_k,t}(Q^{\pi_k}_{r_{\theta_k},t+1}(s,a))$ for any (s,a).*

Proof.

$$
\begin{aligned}
Q^{\pi_{k+1}}_{r_{\theta_k},t}(s,a) &\overset{(a)}{=} r_{\theta_k}(s,a) + \gamma \sum_{a' \in \mathcal{A}} \pi_{k+1,t+1}(a'|s')[Q^{\pi_{k+1}}_{r_{\theta_k},t+1}(s',a') - \beta \log \frac{\pi_{k+1,t+1}(a'|s')}{\pi_{\text{ref},t+1}(a'|s')}] \\
&\overset{(b)}{\geq} r_{\theta_k}(s,a) + \gamma \sum_{a' \in \mathcal{A}} \pi_{k+1,t+1}(a'|s')[Q^{\pi_k}_{r_{\theta_k},t+1}(s',a') - \beta \log \frac{\pi_{k+1,t+1}(a'|s')}{\pi_{\text{ref},t+1}(a'|s')}] \\
&= r_{\theta_k}(s,a) + \gamma\beta \log \Big[\sum_{a' \in \mathcal{A}} \pi_{\text{ref},t+1}(a'|s') \exp(\frac{1}{\beta} Q^{\pi_k}_{r_{\theta_k},t+1}(s',a')) \Big] \\
&= \mathcal{T}_{\theta_k,t}(Q^{\pi_k}_{r_{\theta_k},t+1}(s,a)),
\end{aligned}
$$

where (a) follows equations (2)-(3) in [51] and (b) follows policy improvement theorem (Theorem 4 in [50]). □

Claim 4. *Suppose Assumption 1 holds. For an arbitrary policy π and reward parameters θ_1 and θ_2, it holds that*

$$
\begin{aligned}
|Q^*_{r_{\theta_1},t}(s,a) - Q^*_{r_{\theta_2},t}(s,a)| &\leq \tilde{C}||\theta_1 - \theta_2|| \\
|Q^\pi_{r_{\theta_1},t}(s,a) - Q^\pi_{r_{\theta_2},t}(s,a)| &\leq \tilde{C}||\theta_1 - \theta_2||,
\end{aligned}
$$

for any $(s,a) \in \mathcal{S} \times \mathcal{A}$ and $0 \leq t \leq T$, and $\tilde{C} = CT$ is a positive constant.

Proof. Recall from the proof of Claim 1 that $Z_{r_\theta,t}(s,a) \triangleq \pi_{\text{ref},t}(a|s) \exp(\frac{1}{\beta} Q^*_{r_\theta,t}(s,a))$ and $\nabla_\theta \log Z_{r_\theta,t}(s,a) = \frac{1}{\beta} E^{\pi_{r_\theta}}[\sum_{i=t}^T \gamma^i \nabla_\theta r_\theta(S_i,A_i)|S_t = s, A_t = a]$. Therefore, we have that

$$
\begin{aligned}
\nabla_\theta Q^\pi_{r_\theta,t}(s,a) &= \beta \nabla_\theta \log Z^\pi_{r_\theta,t}(s,a) \\
&= E^\pi[\sum_{i=t}^T \gamma^i \nabla_\theta r_\theta(S_i,A_i)|S_t = s, A_t = a] \\
&\overset{(c)}{\leq} CT,
\end{aligned}
$$

where (c) follows Lemma 4 that $||\nabla_\theta r_\theta(s,a)|| \leq C$. Note that on a convex domain, $|f(x_1) - f(x_2)| \leq L||x_1 - x_2||$ and $||\nabla f(x)|| \leq L$ are equivalent [52, 53], where L is a positive constant.

The action value function can be equivalently reformulated as [50, 37]:

$$
Q^\pi_{r,t}(s,a) = E^\pi \Big[\sum_{i=t}^T \gamma^i r(S_i,A_i) - \beta \log \frac{\pi(A_i|S_i)}{\pi_{\text{ref}}(A_i|S_i)} |S_t = s, A_t = a \Big]. \tag{8}
$$

Therefore, we have that

$$
\begin{aligned}
&\Big| Q^\pi_{r_{\theta_1},t}(s,a) - Q^\pi_{r_{\theta_2},t}(s,a) \Big| \\
&\overset{(d)}{=} \Big| E^\pi \Big[\sum_{i=t}^T \gamma^i r_{\theta_1}(S_i,A_i) - \beta \log \frac{\pi(A_i|S_i)}{\pi_{\text{ref}}(A_i|S_i)} |S_t = s, A_t = a \Big] \\
&- E^\pi \Big[\sum_{i=t}^T \gamma^i r_{\theta_2}(S_i,A_i) - \beta \log \frac{\pi(A_i|S_i)}{\pi_{\text{ref}}(A_i|S_i)} |S_t = s, A_t = a \Big] \Big| \\
&= \Big| E^\pi \Big[\sum_{i=t}^T \gamma^i r_{\theta_1}(S_i,A_i) - r_{\theta_2}(S_i,A_i) |S_t = s, A_t = a \Big] \Big| \\
&\overset{(e)}{\leq} CT||\theta_1 - \theta_2||,
\end{aligned}
$$

where (d) follows (8) and (e) follows Lemma 4. □

Now, we proceed to quantify the convergence of the lower-level policy in Algorithm 1. For any $(s,a) \in \mathcal{S} \times \mathcal{A}$, we know that

$$
\begin{aligned}
&|\log \pi_{k+1,t}(a|s) - \log \pi_{\theta_k,t}(a|s)| \\
&\leq \frac{1}{\beta}(|Q^{\pi_k}_{r_{\theta_k},t}(s,a) - Q^*_{r_{\theta_k},t}(s,a)| + |V^{\pi_k}_{r_{\theta_k},t}(s) - V^*_{r_{\theta_k},t}(s)|) \\
&\overset{(f)}{\leq} \frac{1}{\beta}(|Q^{\pi_k}_{r_{\theta_k},t}(s,a) - Q^*_{r_{\theta_k},t}(s,a)| + \max_{(s,a)\in\mathcal{S}\times\mathcal{A}, 0\leq t\leq T}\{|Q^{\pi_k}_{r_{\theta_k},t}(s,a) - Q^*_{r_{\theta_k},t}(s,a)|\}), \qquad (9)
\end{aligned}
$$

where (f) follows equation (48) in [40].

Now we take a look at the term $|Q^{\pi_k}_{r_{\theta_k},t}(s,a) - Q^*_{r_{\theta_k},t}(s,a)|$:

$$
\begin{aligned}
&|Q^{\pi_k}_{r_{\theta_k},t}(s,a) - Q^*_{r_{\theta_k},t}(s,a)| \leq |Q^{\pi_k}_{r_{\theta_k},t}(s,a) - Q^{\pi_k}_{r_{\theta_{k-1}},t}(s,a)| \\
&+ |Q^{\pi_k}_{r_{\theta_{k-1}},t}(s,a) - Q^*_{r_{\theta_{k-1}},t}(s,a)| + |Q^*_{r_{\theta_{k-1}},t}(s,a) - Q^*_{r_{\theta_k},t}(s,a)| \\
&\overset{(f)}{\leq} \tilde{C}||\theta_k - \theta_{k-1}|| + |Q^{\pi_k}_{r_{\theta_{k-1}},t}(s,a) - Q^*_{r_{\theta_{k-1}},t}(s,a)| + \tilde{C}||\theta_k - \theta_{k-1}|| \\
&= 2\tilde{C}||\theta_k - \theta_{k-1}|| + Q^*_{r_{\theta_{k-1}},t}(s,a) - Q^{\pi_k}_{r_{\theta_{k-1}},t}(s,a) \\
&\overset{(g)}{\leq} 2\tilde{C}||\theta_k - \theta_{k-1}|| + Q^*_{r_{\theta_{k-1}},t}(s,a) - \mathcal{T}_{\theta_{k-1},t}(Q^{\pi_{k-1}}_{r_{\theta_{k-1}},t+1}(s,a)) \\
&= 2\tilde{C}||\theta_k - \theta_{k-1}|| + \mathcal{T}_{\theta_{k-1},t}(Q^*_{r_{\theta_{k-1}},t+1}(s,a)) - \mathcal{T}_{\theta_{k-1},t}(Q^{\pi_{k-1}}_{r_{\theta_{k-1}},t+1}(s,a)) \\
&\overset{(h)}{\leq} 2\tilde{C}||\theta_k - \theta_{k-1}|| + \gamma|Q^*_{r_{\theta_{k-1}},t}(s,a) - Q^{\pi_{k-1}}_{r_{\theta_{k-1}},t}(s,a)|, \qquad (10)
\end{aligned}
$$

where (f) follows Claim 4, (g) follows Claim 3, and (h) follows Claim 2. Therefore, we have that

$$
\begin{aligned}
&\sum_{k=1}^{K} |Q^{\pi_k}_{r_{\theta_k},t}(s,a) - Q^*_{r_{\theta_k},t}(s,a)| \\
&\leq 2\tilde{C}\sum_{k=1}^{K} ||\theta_k - \theta_{k-1}|| + \gamma\sum_{k=1}^{K} |Q^*_{r_{\theta_{k-1}},t}(s,a) - Q^{\pi_{k-1}}_{r_{\theta_{k-1}},t}(s,a)| \\
&= 2\tilde{C}\sum_{k=1}^{K} \alpha_{k-1}||\hat{\nabla}J(\theta_{k-1})|| + \gamma\sum_{k=1}^{K} |Q^*_{r_{\theta_{k-1}},t}(s,a) - Q^{\pi_{k-1}}_{r_{\theta_{k-1}},t}(s,a)| \\
&\overset{(i)}{\leq} 2\tilde{C}\sum_{k=1}^{K} \alpha_{k-1}\frac{2CT^2}{\beta} + \gamma\sum_{k=1}^{K} |Q^*_{r_{\theta_{k-1}},t}(s,a) - Q^{\pi_{k-1}}_{r_{\theta_{k-1}},t}(s,a)|
\end{aligned}
$$

where (i) follows Lemma 5. By subtracting $\gamma\sum_{k=1}^{K} |Q^{\pi_k}_{r_{\theta_k},t}(s,a) - Q^*_{r_{\theta_k},t}(s,a)|$ on both sides, we have that

$$
\begin{aligned}
&(1-\gamma)\sum_{k=1}^{K} |Q^{\pi_k}_{r_{\theta_k},t}(s,a) - Q^*_{r_{\theta_k},t}(s,a)| \\
&\leq \frac{4\tilde{C}CT^2}{\beta}\sum_{k=1}^{K} \alpha_{k-1} + \gamma\Big(|Q^*_{r_{\theta_0},t}(s,a) - Q^{\pi_0}_{r_{\theta_{k-1}},t}(s,a)| - |Q^*_{r_{\theta_K},t}(s,a) - Q^{\pi_K}_{r_{\theta_{k-1}},t}(s,a)|\Big)
\end{aligned}
$$

Therefore, we have that

$$
\begin{aligned}
&\sum_{k=1}^{K} |Q^{\pi_k}_{r_{\theta_k},t}(s,a) - Q^*_{r_{\theta_k},t}(s,a)| \leq \frac{4\tilde{C}CT^2}{\beta(1-\gamma)}\sum_{k=1}^{K} \alpha_{k-1} \\
&+ \frac{\gamma}{(1-\gamma)}\Big(|Q^*_{r_{\theta_0},t}(s,a) - Q^{\pi_0}_{r_{\theta_{k-1}},t}(s,a)| - |Q^*_{r_{\theta_K},t}(s,a) - Q^{\pi_K}_{r_{\theta_{k-1}},t}(s,a)|\Big). \qquad (11)
\end{aligned}
$$

Finally, we have that

$$
\begin{aligned}
&\frac{1}{K}\sum_{k=0}^{K-1}|\log\pi_{k+1,t}(a|s)-\log\pi_{\theta_k,t}(a|s)| \\
&\overset{(j)}{\leq}\frac{1}{\beta K}\sum_{k=1}^{K}(|Q^{\pi_k}_{r_{\theta_k},t}(s,a)-Q^{*}_{r_{\theta_k},t}(s,a)|+\max_{(s,a)\in\mathcal{S}\times\mathcal{A},0\leq t\leq T}\{|Q^{\pi_k}_{r_{\theta_k},t}(s,a)-Q^{*}_{r_{\theta_k},t}(s,a)|\}) \\
&\overset{(k)}{\leq}\frac{8\tilde{C}CT^2}{\beta^2(1-\gamma)K}\sum_{k=1}^{K}k^{1/2} \\
&+\frac{2\gamma}{(1-\gamma)K}\Big(|Q^{*}_{r_{\theta_0},t}(s,a)-Q^{\pi_0}_{r_{\theta_{k-1}},t}(s,a)|-|Q^{*}_{r_{\theta_K},t}(s,a)-Q^{\pi_K}_{r_{\theta_{k-1}},t}(s,a)|\Big) \\
&=O(1/\sqrt{K})+O(1/K),
\end{aligned}
$$

where (j) follows (9) and (k) follows (11).

A.6 Proof of Theorem 1

$$
\begin{aligned}
J(\theta_{k+1})&\overset{(a)}{\geq}J(\theta_k)+\langle\nabla J(\theta_k),\theta_k-\theta_{k-1}\rangle-\frac{C_J}{2}||\theta_{k+1}-\theta_k||^2 \\
&=J(\theta_k)+\alpha_k\langle\nabla J(\theta_k),\hat{\nabla}J(\theta_k)\rangle-\frac{C_J\alpha_k^2}{2}||\hat{\nabla}J(\theta_k)||^2 \\
&=J(\theta_k)+\alpha_k\langle\nabla J(\theta_k),\hat{\nabla}J(\theta_k)-\nabla J(\theta_k)\rangle+\alpha_k||\nabla J(\theta_k)||^2-\frac{C_J\alpha_k^2}{2}||\hat{\nabla}J(\theta_k)||^2 \\
&\overset{(b)}{\geq}J(\theta_k)+\alpha_k\langle\nabla J(\theta_k),\hat{\nabla}J(\theta_k)-\nabla J(\theta_k)\rangle+\alpha_k||\nabla J(\theta_k)||^2-\frac{CC_JT^2\alpha_k^2}{\beta}.
\end{aligned}
$$

where (a) follows Lemma 6 and (b) follows Lemma 5. Take expectation on both sides, we have that

$$
E[J(\theta_{k+1})]\geq E[J(\theta_k)]+\alpha_kE[\langle\nabla J(\theta_k),\hat{\nabla}J(\theta_k)-\nabla J(\theta_k)\rangle]+\alpha_kE[||\nabla J(\theta_k)||^2]-\frac{CC_JT^2\alpha_k^2}{\beta}.
$$

Now, we take a look at the term

$$
\begin{aligned}
&E[\langle\nabla J(\theta_k),\hat{\nabla}J(\theta_k)-\nabla J(\theta_k)\rangle] \\
&\overset{(c)}{=}E\Big[\langle\nabla J(\theta_k),E_{x\sim\mathcal{D}}E^{\pi_{k+1}}\Big[\sum_{t=0}^{T}\gamma^t\nabla_\theta\Big(\log Z_{r_\theta,t}(S_t,A_t)-\log Z_{r_\theta,t}(S_t)\Big)r_{\text{corr}}(S_T,A_T)|S_0=x\Big] \\
&-E_{x\sim\mathcal{D}}E^{\pi_{r_{\theta_k}}}\Big[\sum_{t=0}^{T}\gamma^t\nabla_\theta\Big(\log Z_{r_\theta,t}(S_t,A_t)-\log Z_{r_\theta,t}(S_t)\Big)r_{\text{corr}}(S_T,A_T)|S_0=x\Big]\rangle\Big] \\
&\overset{(d)}{\leq}\frac{2CT^2}{\beta}\Big|\Big|E_{x\sim\mathcal{D}}E^{\pi_{k+1}}\Big[\sum_{t=0}^{T}\gamma^t\nabla_\theta\Big(\log Z_{r_\theta,t}(S_t,A_t)-\log Z_{r_\theta,t}(S_t)\Big)r_{\text{corr}}(S_T,A_T)|S_0=x\Big] \\
&-E_{x\sim\mathcal{D}}E^{\pi_{r_{\theta_k}}}\Big[\sum_{t=0}^{T}\gamma^t\nabla_\theta\Big(\log Z_{r_\theta,t}(S_t,A_t)-\log Z_{r_\theta,t}(S_t)\Big)r_{\text{corr}}(S_T,A_T)|S_0=x\Big]\rangle\Big]\Big|\Big| \\
&\overset{(e)}{\leq}\frac{8C^2T^3}{\beta}\sqrt{|\mathcal{S}|\cdot|\mathcal{A}|}E\Big[||Q^{\pi_k}_{r_{\theta_k},t}-Q^{*}_{r_{\theta_k},t}||_\infty\Big],
\end{aligned}
$$

where (c) follows (5), (d) follows Lemma 5, (e) follows equation (64) in [40]. Note that (e) (or equation (64) in [40]) depends on an assumption that the Markov chain induced by the transition function P is ergodic. Therefore, we have that

$$
\begin{aligned}
&E[J(\theta_{k+1})] \\
&\geq E[J(\theta_k)]-\frac{8C^2T^3\alpha_k}{\beta}\sqrt{|\mathcal{S}|\cdot|\mathcal{A}|}E\Big[||Q^{\pi_k}_{r_{\theta_k},t}-Q^{*}_{r_{\theta_k},t}||_\infty\Big]+\alpha_kE[||\nabla J(\theta_k)||^2]-\frac{CC_JT^2\alpha_k^2}{\beta}
\end{aligned}
$$

$$
\begin{aligned}
&\Rightarrow \alpha_k E[||\nabla J(\theta_k)||^2] \\
&\leq E[J(\theta_{k+1})] - E[J(\theta_k)] + \frac{8C^2T^3\alpha_k}{\beta}\sqrt{|\mathcal{S}|\cdot|\mathcal{A}|}E\Big[||Q^{\pi_k}_{r_{\theta_k},t} - Q^*_{r_{\theta_k},t}||_\infty\Big] + \frac{CC_JT^2\alpha_k^2}{\beta} \\
&\Rightarrow \sum_{k=0}^{K-1}\frac{1}{\sqrt{K}}E[||\nabla J(\theta_k)||^2] \leq \sum_{k=0}^{K-1}\alpha_k E[||\nabla J(\theta_k)||^2] \\
&\leq E[J(\theta_K)] - E[J(\theta_0)] + \frac{8C^2T^3\alpha_k}{\beta}\sqrt{|\mathcal{S}|\cdot|\mathcal{A}|}\sum_{k=0}^{K-1}E\Big[||Q^{\pi_k}_{r_{\theta_k},t} - Q^*_{r_{\theta_k},t}||_\infty\Big] + \sum_{k=0}^{K-1}\frac{CC_JT^2\alpha_k^2}{\beta} \\
&\Rightarrow \frac{1}{K}\sum_{k=0}^{K-1}E[||\nabla J(\theta_k)||^2] \\
&\leq \frac{E[J(\theta_K)] - E[J(\theta_0)]}{\sqrt{K}} + \frac{8C^2T^3\sqrt{|\mathcal{S}|\cdot|\mathcal{A}|}}{\beta\sqrt{K}}\sum_{k=0}^{K-1}\alpha_k E\Big[||Q^{\pi_k}_{r_{\theta_k},t} - Q^*_{r_{\theta_k},t}||_\infty\Big] + \frac{CC_JT^2\log K}{\sqrt{K}} \\
&= O(\frac{1}{\sqrt{K}}) + O(\frac{\log K}{\sqrt{K}}) + \frac{8C^2T^3\sqrt{|\mathcal{S}|\cdot|\mathcal{A}|}}{\beta\sqrt{K}}\sum_{k=0}^{K-1}\alpha_k E\Big[||Q^{\pi_k}_{r_{\theta_k},t} - Q^*_{r_{\theta_k},t}||_\infty\Big]. \qquad (12)
\end{aligned}
$$

Now we take a look at the last term in the above formula:

$$
\begin{aligned}
&\sum_{k=1}^{K-1}\alpha_k E\Big[||Q^{\pi_k}_{r_{\theta_k},t} - Q^*_{r_{\theta_k},t}||_\infty\Big] \overset{(f)}{\leq} \sum_{k=1}^{K-1}2\tilde{C}\alpha_k E[||\theta_k - \theta_{k-1}||] + \gamma\alpha_k E[||Q^*_{r_{\theta_{k-1}},t} - Q^{\pi_{k-1}}_{r_{\theta_{k-1}},t}||_\infty] \\
&= \sum_{k=1}^{K-1}2\tilde{C}\alpha_k\alpha_{k-1}E[||\hat{\nabla}J(\theta_{k-1})||] + \gamma\alpha_k E[||Q^*_{r_{\theta_{k-1}},t} - Q^{\pi_{k-1}}_{r_{\theta_{k-1}},t}||_\infty] \\
&\overset{(g)}{\leq} \sum_{k=1}^{K-1}2\tilde{C}\alpha_k\alpha_{k-1}\frac{2CT^2}{\beta} + \gamma\alpha_k E[||Q^*_{r_{\theta_{k-1}},t} - Q^{\pi_{k-1}}_{r_{\theta_{k-1}},t}||_\infty],
\end{aligned}
$$

where (f) follows (10) and (g) follows Lemma 5. Subtracting $\sum_{k=1}^{K-1}\gamma\alpha_k E\Big[||Q^{\pi_k}_{r_{\theta_k},t} - Q^*_{r_{\theta_k},t}||_\infty\Big]$ on both sides, we have that

$$
\begin{aligned}
&(1-\gamma)\sum_{k=1}^{K-1}\alpha_k E\Big[||Q^{\pi_k}_{r_{\theta_k},t} - Q^*_{r_{\theta_k},t}||_\infty\Big] \\
&\leq \frac{4\tilde{C}CT^2}{\beta}\sum_{k=1}^{K-1}\alpha_k\alpha_{k-1} + \gamma\alpha_1 E\Big[||Q^{\pi_1}_{r_{\theta_1},t} - Q^*_{r_{\theta_1},t}||_\infty\Big] - \gamma\alpha_{K-1}E\Big[||Q^{\pi_{K-1}}_{r_{\theta_{K-1}},t} - Q^*_{r_{\theta_{K-1}},t}||_\infty\Big]. \qquad (13)
\end{aligned}
$$

Plugging (13) back to (12), we have that

$$
\begin{aligned}
&\frac{1}{K}\sum_{k=0}^{K-1}E[||\nabla J(\theta_k)||^2] \\
&\leq O(\frac{1}{\sqrt{K}}) + O(\frac{\log K}{\sqrt{K}}) \\
&+ \frac{8C^2T^3\sqrt{|\mathcal{S}|\cdot|\mathcal{A}|}}{\beta\sqrt{K}}\Big[\alpha_0 E\Big[||Q^{\pi_0}_{r_{\theta_0},t} - Q^*_{r_{\theta_0},t}||_\infty \\
&+ \sum_{k=1}^{K-1}2\tilde{C}\alpha_k\alpha_{k-1}\frac{2CT^2}{\beta} + \gamma\alpha_k E[||Q^*_{r_{\theta_{k-1}},t} - Q^{\pi_{k-1}}_{r_{\theta_{k-1}},t}||_\infty]\Big] \\
&\leq O(\frac{1}{\sqrt{K}}) + O(\frac{\log K}{\sqrt{K}}) \\
&+ \frac{8C^2T^3\sqrt{|\mathcal{S}|\cdot|\mathcal{A}|}}{\beta\sqrt{K}}\Big[\alpha_0 E\Big[||Q^{\pi_0}_{r_{\theta_0},t} - Q^*_{r_{\theta_0},t}||_\infty
\end{aligned}
$$

$$+\sum_{k=1}^{K-1} 2\tilde{C}\alpha_k^2 \frac{2CT^2}{\beta} + \gamma\alpha_k E[||Q^*_{r_{\theta_{k-1}},t} - Q^{\pi_{k-1}}_{r_{\theta_{k-1}},t}||_\infty]\Big]$$
$$= O(\frac{1}{\sqrt{K}}) + O(\frac{\log K}{\sqrt{K}}).$$

B Experiment details

All experiments use the same configuration across models and benchmarks. Each iteration samples a batch of 80 prompts, and for each prompt we generate $m = 10$ responses. We run for 330 iterations. Both policy and reward parameters are optimized using AdamW with constant learning rates of 10^{-6}. While in theory the reward learning rate needs to be diminishing, in practice the constant step size can already lead to good performance. The KL coefficient is $\beta = 0.05$. All results are computed from 3 random seeds. Experiments are conducted on 8 A100 80G GPUs.

B.1 SQL query generation

For SQL benchmarks, we filter out prompts whose length exceeds 6,144 tokens. The response length is restricted within 2048 tokens. We use the same prompt template as the one in OmiSQL [12]:

Prompt for SQL query generation

```
Task Overview:  You are a data science expert.  Below, you are
provided with a database schema and a natural language question.  Your
task is to understand the schema and generate a valid SQL query to
answer the question.

Database Engine:
SQLite

Database Schema:
{db_details}
This schema describes the database's structure, including tables,
columns, primary keys, foreign keys, and any relevant relationships
or constraints.

Question:
{question}

Instructions:
- Make sure you only output the information that is asked in the
question.  If the question asks for a specific column, make sure to
only include that column in the SELECT clause, nothing more.
- The generated query should return all of the information asked in
the question without any missing or extra information.
- Before generating the final SQL query, please think through the
steps of how to write the query.

Output Format:
In your answer, please enclose the generated SQL query in a code
block:
"'
- Your SQL query
"'

Take a deep breath and think step by step to find the correct SQL
query.
```

In the following context, we provide sampled examples on both BIRD and spider datasets. From the examples, we can see that correctness reward and composite reward can generate almost correct queries but the tiny errors make the entire queries incorrect. RESHAPE provides token-level reward and can penalize the tiny error and thus leads to correct queries.

Table 4: Example of generated queries on a BIRD question with Qwen2.5-Coder-3B-Instruct using GRPO as policy optimizer.

Question

```
Charter schools refers to 'Charter School (Y/N)' =
1 in the frpm Please list the phone numbers of the
direct charter-funded schools that are opened after
2000/1/1.
```

Base model

```
1 SELECT Phone
2 FROM frpm
3 WHERE Charter = 1
4 AND Charter Funding Type = 'Directly funded
  '
5 AND OpenDate >= '2000-01-01';
```

GRPO+Correctness reward

```
1 SELECT s.Phone
2 FROM schools s
3 JOIN frpm f ON s.CDSCode = f.CDSCode
4 WHERE f.`Charter School (Y/N)` = 1
5 AND s.FundingType = 'Directly funded'
6 AND s.OpenDate > '2000-01-01';
```

GRPO+Composite reward

```
1 SELECT s.Phone
2 FROM schools s
3 JOIN frpm f ON s.CDSCode = f.CDSCode
4 WHERE f.`Charter School (Y/N)` = 1
5 AND s.FundingType = f.`Charter Funding Type`
6 AND s.OpenDate > '2000-01-01';
```

GRPO+PRIME

```
1 SELECT s.Phone
2 FROM schools s
3 JOIN frpm f ON s.CDSCode = f.CDSCode
4 WHERE f.`Charter School (Y/N)` = 1
5 AND f.`Charter Funding Type` = 'Directly
  funded'
6 AND s.OpenDate > '2000-01-01';
```

GRPO+RESHAPE

```
1 SELECT s.Phone
2 FROM schools s
3 JOIN frpm f ON s.CDSCode = f.CDSCode
4 WHERE f.`Charter School (Y/N)` = 1
5 AND f.`Charter Funding Type` = 'Directly
  funded'
6 AND s.OpenDate > '2000-01-01';
```

In the example of Table 4, the query generated by the base model is not executable. The queries generated under correctness reward, composite reward, and RESHAPE are all executable, however only the query under RESHAPE is correct. We can see that the queries under correctness reward and composite reward are both almost correct and they both have small local errors at line 5. The correct condition is `f.`Charter Funding Type` = 'Directly funded'` but the query under correctness reward writes `s.FundingType = 'Directly funded'` and the query under composite reward writes `s.FundingType = f.`Charter Funding Type``. They both write wrong conditions but the wrong conditions are similar to the correct condition. However, both the correctness reward and composite reward penalize the entire query and thus it is difficult for the model to locate the error, while PRIME and RESHAPE can penalize local errors and thus the generated query can correct this local error and become correct.

Table 5: Example of generated queries on a BIRD question with deepseek-coder-6.7b-instruct using ReMax as policy optimizer.

Question

What is the unabbreviated mailing street address of the school with the highest FRPM count for K-12 students?

Base model

```
             SELECT MailStreet, MailStrAbr, MailCity,
MailZip, MailState
             FROM schools
             JOIN frpm ON schools.CDSCode = frpm.CDSCode
             WHERE SchoolType = 'K-12 Schools (Public)'
             ORDER BY `FRPM Count (K-12)` DESC \nLIMIT
1;
```

ReMax+Correctness reward

```
            SELECT MailStreet
            FROM schools
            WHERE CDSCode IN (
            SELECT CDSCode
            FROM frpm
            WHERE `School Type` = 'K-12 Schools (Public)
'
            AND `FRPM Count (K-12)` = (
            SELECT MAX(`FRPM Count (K-12)`)
            FROM frpm
            WHERE `School Type` = 'K-12 Schools (Public)
'));
```

ReMax+Composite reward

```
             SELECT MailStreet
             FROM schools
             WHERE CDSCode = (
             SELECT CDSCode
             FROM frpm
             WHERE `School Type` = 'K-12 Schools (Public
)'
             AND `FRPM Count (K-12)` = (
             SELECT MAX(`FRPM Count (K-12)`)
             FROM frpm
             WHERE `School Type` = 'K-12 Schools (Public
)'));
```

ReMax+PRIME

```
            SELECT MailStreet
            FROM schools
            WHERE CDSCode IN (
            SELECT CDSCode
            FROM frpm
            WHERE `School Type` ='K-12 Schools (Public)'
             AND
            `FRPM Count (K-12)` = (
            SELECT MAX(`FRPM Count (K-12)`)
            FROM frpm));
```

ReMax+RESHAPE

```
            SELECT MailStreet
            FROM schools
            WHERE CDSCode IN (
            SELECT CDSCode
            FROM frpm
            WHERE `FRPM Count (K-12)` = (
            SELECT MAX(`FRPM Count (K-12)`)
            FROM frpm));
```

In Table 5, only the RESHAPE query is correct and the other queries are incorrect as they use the wrong condition `'School Type' = 'K-12 Schools (Public)'`.

In Table 6, only the RESHAPE query is correct and the other queries are incorrect. For example, the query generated by composite reward is wrong because `Order_Items` does not have `customer_id`. The correct property is `order_id`.

Table 6: Example of generated queries on a spider question with Qwen2.5-Coder-7B-Instruct using RLOO as policy optimizer.

Question	`What are the ids, first and last names of the customers who have ordered more than twice and have bought at least 3 items?`

Base model

```
            SELECT c.customer_id, c.customer_first_name,
 c.customer_last_name
            FROM
            Customers c
            JOIN
            Orders o ON c.customer_id = o.customer_id
            JOIN
            Order_Items oi ON o.order_id = oi.order_id
            GROUP BY
            c.customer_id, c.customer_first_name, c.
customer_last_name
            HAVING
            COUNT(o.order_id) > 2
            AND COUNT(oi.order_item_id) >= 3;
```

RLOO+Correctness reward

```
            SELECT c.customer_id, c.customer_first_name,
 c.customer_last_name
            FROM Customers c
            JOIN Orders o ON c.customer_id = o.
customer_id
            GROUP BY c.customer_id
            HAVING COUNT(o.order_id) > 2
            AND (
            SELECT SUM(CASE WHEN oi.
order_item_status_code = 'Delivered' THEN 1 ELSE 0
END)
            FROM Order_Items oi
            JOIN Orders o2 ON oi.order_id = o2.order_id
            WHERE o2.customer_id = c.customer_id) >= 3;
```

RLOO+Composite reward

```
    SELECT c.customer_id, c.customer_first_name, c.
customer_last_name
    FROM Customers c
    JOIN Orders o ON c.customer_id = o.customer_id
    GROUP BY c.customer_id
    HAVING COUNT(o.order_id) > 2
    INTERSECT
    SELECT c.customer_id, c.customer_first_name, c.
customer_last_name
    FROM Customers c
    JOIN Order_Items oi ON c.customer_id = oi.
customer_id
    GROUP BY c.customer_id
    HAVING COUNT(oi.order_item_id) >= 3;
```

RLOO+PRIME

```
SELECT c.customer_id, c.customer_first_name, c.
customer_last_name
FROM Customers c
JOIN Orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_id
HAVING COUNT(o.order_id) > 2
INTERSECT
SELECT c.customer_id, c.customer_first_name, c.
customer_last_name
FROM Customers c
JOIN Order_Items oi ON c.customer_id = oi.
customer_id
GROUP BY c.customer_id
HAVING COUNT(oi.order_item_id) >= 3;
```

RLOO+RESHAPE

```
SELECT c.customer_id, c.customer_first_name, c.
customer_last_name
FROM Customers c
JOIN Orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_id
HAVING COUNT(o.order_id) > 2
AND EXISTS (
SELECT 1
FROM Orders o2
JOIN Order_Items oi ON o2.order_id = oi.order_id
WHERE o2.customer_id = c.customer_id
GROUP BY o2.customer_id
HAVING COUNT(oi.order_item_id) >= 3);
```

From the above qualitative example, we can see that the token-level rewards (PRIME and RESHAPE) can help correct local errors, while RESHAPE can correct local errors in more cases than PRIME.

B.1.1 Comparison to PPO

While PPO's critic and RESHAPE may appear structurally similar, they differ fundamentally in three aspects: learning objective, optimization structure, and update rule.

Distinction in learning objective. A clear way to separate RESHAPE from PPO is to start from what is being learned. In PPO, the critic learns a value function V^π, which is an estimate of the expected cumulative reward under the current policy. This is fundamentally an evaluation problem: given a fixed reward function r, the critic approximates how good each state is. In contrast, RESHAPE learns a reward function r_θ itself. This is a supervision design problem: instead of estimating returns under a given objective, it learns what signal the policy should optimize. Thus, value learning is policy-dependent and descriptive (it describes the current policy's behavior), whereas reward learning is prescriptive and can shape how the policy should behave.

This distinction leads to the effect that RESHAPE changes the optimization objective, while PPO does not. Specifically, PPO always optimizes the same underlying objective:

$$\max_\pi E^\pi_{x\sim\mathcal{D}}\Big[\sum_{t=0}^{T}\gamma^t r_{\text{corr}}(S_t, A_t)|S_0 = x\Big],$$

where the KL regularization is omitted. The critic only affects how the gradient is estimated (through the advantage function), not what is being optimized.

In contrast, RESHAPE learns a shaping reward and modify the optimization objective to

$$\max_\pi E^\pi_{x\sim\mathcal{D}}\Big[\sum_{t=0}^{T}\gamma^t r_\theta(S_t, A_t)|S_0 = x\Big],$$

where r_θ is optimized and thus dynamically changes. This means RESHAPE modifies the learning objective itself, effectively reshaping the optimization landscape, whereas PPO leaves the objective unchanged and only improves optimization efficiency.

Distinction in optimization structure. A second important distinction lies in the optimization structure. RESHAPE is a bilevel optimization framework, where the upper level optimizes the shaping reward and the lower level optimizes the policy. In contrast, PPO is only to optimize the policy. In fact, PPO can serve as the lower-level policy optimization algorithm of RESHAPE. In other words, RESHAPE can integrate PPO in the sense that the upper level optimizes the shaping reward and the lower level uses PPO to optimize the corresponding policy under the shaping reward r_θ. We include an empirical evaluation to demonstrate this in the following context.

Distinction in update rule. Finally, the update rules for PPO's critic and RESHAPE's shaping reward differ significantly. In RESHAPE, the shaping reward is optimized to maximize the downstream RL objective: $J(\theta) = E^{\pi_{r_\theta}}_{x\sim\mathcal{D}}[\sum_{t=0}^{T}\gamma^t r_{\text{corr}}(S_t, A_t)|S_0 = x)]$. From (5), the corresponding gradient takes the form of a policy gradient with respect to the reward parameters:

$$\nabla J(\theta) = E_{x\sim\mathcal{D}} E^{\pi_{r_\theta}}\Big[\sum_{t=0}^{T}\gamma^t \nabla_\theta \log \pi_{r_\theta}(A_t|S_t) Q^{\pi_{r_\theta}}_{r_{\text{corr}}}(S_t, A_t)|S_0 = x\Big],$$

where $Q^{\pi_{r_\theta}}_{r_{\text{corr}}}(s_t, a_t) = r_{\text{corr}}(s_t, a_t) + E^{\pi_{r_\theta}}[\sum_{i=t+1}^{T}\gamma^i r_{\text{corr}}(S_i, A_i)|S_t = s_t, A_t = a_t]$ is the action-value function. In PPO, the critic V_ϕ parameterized by ϕ is trained to approximate the value function under the current policy π by minimizing a regression loss: $\min_\phi \; E^\pi[(V_\phi(s_t) - \hat{R}_t)^2]$, where $\hat{R}_t = \sum_{i=t}^{T}\gamma^{i-t} r_{\text{corr}}(s_i, a_i)$ is the empirical return. The gradient for PPO update is

$$\nabla_\phi E^\pi[(V_\phi(s_t) - \hat{R}_t)^2].$$

The key difference is that PPO critic update is to minimize a supervised regression loss to fit returns under the current policy. In contrast, the RESHAPE reward is updated to maximize the downstream RL objective. This distinction leads to different gradients and different update rules.

In addition, we conduct an experiment using PPO as the policy optimizer. The results in Table 7 demonstrate that RESHAPE still outperforms the baselines when using PPO as the policy optimizer.

Table 7: Performance comparison when the policy optimizer is PPO.

Model	Reward	BIRD	Spider
Qwen2.5-Coder-3B-Instruct	Outcome	52.17 ± 0.78	82.98 ± 1.09
	Composite	54.36 ± 0.62	83.37 ± 0.55
	PRIME	55.67 ± 0.75	84.56 ± 0.64
	RESHAPE	$\mathbf{57.12 \pm 0.73}$	$\mathbf{87.98 \pm 0.84}$
deepseek-coder-6.7b-instruct	Outcome	52.03 ± 0.65	82.55 ± 0.62
	Composite	53.59 ± 0.68	83.57 ± 1.02
	PRIME	54.62 ± 0.73	84.36 ± 0.75
	RESHAPE	$\mathbf{57.05 \pm 0.51}$	$\mathbf{87.42 \pm 0.52}$

B.1.2 RL training from base model

We provide results of direct RL training from base models for SQL benchmarks BIRD and spider. Note that we do not provide the results for the Cypher benchmark because the Cypher benchmark is too complicated where the performance of the base model is less than 5%, and direct RL training from the base model does not work. In particular, the Cypher benchmark has a very long ontology (more than 64k tokens) and it is difficult for models with size smaller than 7B to comprehend the ontology and generate correct queries. Therefore, we need to use SFT to first let the model acquire the knowledge about the ontology. The results in Table 8 demonstrate that RESHAPE outperforms the baselines.

Table 8: Performance comparison for RL training from the base models.

Model	Policy Optimizer	Reward Configuration	BIRD	Spider
Qwen2.5-Coder-3B-Instruct	No training	N/A	35.66 ± 0.78	68.23 ± 0.83
	GRPO	Correctness reward	50.69 ± 0.71	80.23 ± 0.79
		Composite reward	51.14 ± 0.62	81.23 ± 0.76
		PRIME	52.88 ± 0.87	82.74 ± 0.65
		RESHAPE	$\mathbf{54.22 \pm 0.74}$	$\mathbf{84.89 \pm 0.65}$
	ReMax	Correctness reward	50.15 ± 0.85	80.82 ± 1.01
		Composite reward	50.78 ± 0.86	82.29 ± 1.02
		PRIME	51.84 ± 0.85	83.67 ± 0.97
		RESHAPE	$\mathbf{53.67 \pm 0.69}$	$\mathbf{86.82 \pm 0.85}$
	RLOO	Correctness reward	51.13 ± 0.69	81.09 ± 0.57
		Composite reward	52.77 ± 0.72	82.62 ± 0.81
		PRIME	53.07 ± 0.74	83.68 ± 1.04
		RESHAPE	$\mathbf{56.13 \pm 0.51}$	$\mathbf{86.21 \pm 0.74}$
deepseek-coder-6.7b-instruct	No training	N/A	40.55 ± 1.04	69.54 ± 1.26
	GRPO	Correctness reward	51.05 ± 0.84	81.94 ± 1.15
		Composite reward	52.91 ± 0.67	83.28 ± 0.89
		PRIME	53.05 ± 0.75	84.65 ± 0.85
		RESHAPE	$\mathbf{55.61 \pm 0.77}$	$\mathbf{86.31 \pm 0.76}$
	ReMax	Correctness reward	48.22 ± 1.02	79.17 ± 0.89
		Composite reward	49.09 ± 0.62	82.01 ± 0.74
		PRIME	50.76 ± 0.85	83.42 ± 1.05
		RESHAPE	$\mathbf{53.72 \pm 0.63}$	$\mathbf{86.12 \pm 0.58}$
	RLOO	Correctness reward	47.11 ± 1.06	75.19 ± 0.98
		Composite reward	48.16 ± 1.07	80.22 ± 0.76
		PRIME	50.23 ± 0.64	79.55 ± 0.64
		RESHAPE	$\mathbf{52.80 \pm 0.68}$	$\mathbf{85.56 \pm 0.54}$
Qwen2.5-Coder-7B-Instruct	No training	N/A	47.98 ± 0.75	82.02 ± 0.92
	GRPO	Correctness reward	55.49 ± 0.93	85.42 ± 0.74
		Composite reward	56.82 ± 0.79	85.21 ± 0.87
		PRIME	57.24 ± 0.98	96.18 ± 0.73
		RESHAPE	$\mathbf{59.08 \pm 0.71}$	$\mathbf{88.62 \pm 0.83}$
	ReMax	Correctness reward	54.69 ± 1.13	84.40 ± 1.02
		Composite reward	54.58 ± 0.85	$\mathbf{87.29 \pm 0.84}$
		PRIME	55.29 ± 0.56	86.12 ± 1.23
		RESHAPE	$\mathbf{57.66 \pm 0.82}$	87.04 ± 1.05
	RLOO	Correctness reward	54.06 ± 0.73	83.22 ± 1.07
		Composite reward	54.88 ± 0.83	84.09 ± 0.52
		PRIME	55.48 ± 0.74	85.12 ± 0.83
		RESHAPE	$\mathbf{57.93 \pm 0.75}$	$\mathbf{87.45 \pm 0.62}$

B.2 Cypher query generation

The data collection pipeline. We first collect real-world user questions on finance (e.g., what is Amazon stock price?) and sports (e.g., aaron judge stats). We partition the collected questions into three sets: SFT training data, RL training data, and test data. For each question in SFT training data, we (1) record the corresponding query time (i.e., the time when the user asked the question), (2) detect/link the entities in the graphs corresponding to the mentions in the question, (3) search the Internet to get a reference answer, and (4) use DeepSeek-R1 to generate a query and the query is then executed to retrieve information from the graph. We use Claude 4.5 to judge whether the retrieved information is correct or not and we accumulate all the prompt-query pair as SFT data if the retrieved information of the query is labeled as correct (including correct and complete, and correct and incomplete) by Claude 4.5. The prompt template for Claude 4.5 is provided below. We first perform SFT instead of training with RL from scratch (like we did for BIRD and spider) because the created benchmark is too complex for base models to generate correct queries. The prompt template for Cypher query generation is provided below.

For each question in the RL training data and test data, we (1) record the corresponding query time, (2) detect/link the entities in the graphs corresponding to the mentions in the question, and (3) search the Internet to get a reference answer. We use Claude 4.5 to judge whether the generated queries are correct or not by comparing their retrieved information to the reference answers.

Prompt for Claude 4.5

```
You are a knowledgeable expert in sports and finance.
<instructions> Your task is to annotate knowledge graph responses vs
question asked.
Response can be annotated with one of following (use only one of
options):
1.  [Correct and Complete]
2.  [Correct and Incomplete]
3.  [Incorrect]

A correct response should be informative and provide valid information
to the user that directly addresses the question being asked.  The
correctness of the answer to the question can be evaluated based on
following factors:
* Relevance:  A correct answer should provide information that is
directly related to the topic or subject matter of the question.
* Coverage:  A correct answer should provide a sufficient level of
detail or coverage to adequately address the question and provide
useful information to the user.
* Accurate:  A correct answer should be consistent with common
knowledge (in most cases), known facts or other credible sources
supported by a web page.  (If you use a search engine, the web page
should be saved for evidence.)
* Timeliness:  The correctness of an answer may be affected by the
current time, location, or other contextual factors.
* Inferred answer:  If the answer to the question can be inferred from
the response, we deem it as correct.

A complete response should:
* Addresses all parts of the question or problem, leaving no component
unanswered.
* Provides sufficient information to fully resolve the inquiry or task
at hand.
* Provides context or background information when necessary for full
understanding.

You will be given:
* Question in <question></question> tags
* Response in <response></response> tags
* Question context in <context></context> tags
* Question date in <date></date> tags
* Ground truth information in <truth></truth> tags - you can use the
to validate response correctness

Explain your decision and provide concise answer.
</instructions>
```

The Cypher benchmark has 200K user questions. While it is infeasible to manually annotate all 200K questions, we conduct human validation on 7K randomly sampled examples. The agreement rate between Claude and human judgments is 88%, indicating that Claude provides a reasonably reliable proxy for correctness. The SFT dataset (15K Sports + 15K Finance) and RL dataset (5K Sports + 5K Finance) are constructed separately from the evaluation sets (10K Sports + 10K Finance). Note that in the paper, we mention that we split the data into 1:1 SFT set and RL set. For the Cypher benchmark, the RL set is further split into 1:2 RL set and evaluation set. The SQL benchmarks have separate test sets so that we do not need to further split the RL set. To address the concern of overfitting to the judge pipeline, we sample 100 sports data and 100 finance data and provide Claude judge and human judge performance for Qwen2.5-Coder-3B-Instruct+GRPO.

One may be concerned about overfitting to the judge pipeline. To address this concern, we sample 100 sports data and 100 finance data and provide Claude judge and human judge performance for Qwen2.5-Coder-3B-Instruct+GRPO. The results in Table 9 show that RESHAPE improves the baselines under both Claude judge and huam judge.

Table 9: Human-Claude alignment.

Judge	Reward Configuration	Sports	Finance
Claude	Correctness	37	52
	Composite	37	54
	PRIME	38	54
	RESHAPE	40	58
Human	Correctness	40	55
	Composite	41	57
	PRIME	41	58
	RESHAPE	43	62

Prompt for Cypher query generation

```
Please generate a Cypher Query to answer the user question with
the provided question domain, entity linking results and timestamp.
Carefully make use of the given entity information and select the
most relevant entity by considering both the confidence score and the
context of the user's question.  Prioritize entities that directly
relate to key terms in the question.  The Cypher Query must include
the selected entity and should be executable.  Decide whether to
include the query timestamp based on its relevance to the user's
request.

Domain:  finance/sports

Entity Linking Results:  {entity_linking_results}

User question:  {query_text}

Query timestamp:  {query_time}
```

The maximum prompt length is 5120 and the maximum response length is 2048. While the composite reward proposed in [5] is originally designed for SQL queries, we adapt it to the Cypher query setting. Specifically, the format reward verifies whether a query contains the mandatory MATCH and RETURN clauses; the execution reward evaluates whether a query is executable; and the length reward prefers shorter responses.

B.3 Computation cost

Table 3 compares the per-iteration computation time on Spider. Since all methods have the same number of iterations, this also reflects the aggregated time. Compared to standard RL, the additional cost of RESHAPE comes from the reward update. Importantly, RESHAPE does not require additional rollouts: the reward update reuses the same sampled trajectories as the policy update. Therefore, the overhead is limited to extra forward and backward passes on shared data. This overhead is modest because rollout generation dominates the overall training cost for large LLMs.

In addition, the memory overhead of the shaping reward can be reduced. The reward model does not need to match the scale of the policy model. For example, the policy can be a 7B model while the reward can be a 1B model, since reward estimation is typically less demanding than generation.

C Ablation Studies

In this section, we conduct ablation studies on the rollout number m, KL coefficient β, lower-level optimization steps, and reward parameterization.

Ablation on the rollout number m. We include the results of RESHAPE under different numbers m of rollout samples (5,10,15,20) for BIRD. We consider different combinations of base models and base RL algorithms.

Table 10: Ablation on the rollout number m.

Model+RL	5	10	15	20
Qwen2.5-Coder-3B-Instruct+GRPO	54.87 ± 0.69	55.49 ± 0.63	55.88 ± 0.57	56.02 ± 0.76
deepseek-coder-6.7b-instruct+ReMax	52.22 ± 0.77	54.40 ± 0.67	54.82 ± 0.78	54.91 ± 0.81
Qwen2.5-Coder-7B-Instruct+RLOO	56.14 ± 0.93	58.05 ± 0.66	58.42 ± 0.77	58.61 ± 0.94

The results in Table 10 show that increasing m from 5 to 10 leads to noticeable improvements. However, beyond $m \geq 10$, the gains become marginal. These results indicate that RESHAPE achieves stable performance once a moderate number of samples is used.

Ablation on the KL coefficient β. We include the results of RESHAPE under different KL coefficient $\beta = 0.01, 0.05, 0.10, 0.15$ for Spider.

Table 11: Ablation on the KL coefficient β.

	0.01	0.05	0.10	0.15
Qwen2.5-Coder-3B-Instruct+GRPO	84.65 ± 0.67	86.02 ± 0.61	85.88 ± 1.14	84.24 ± 0.66
deepseek-coder-6.7b-instruct+ReMax	86.21 ± 0.68	87.04 ± 0.62	87.21 ± 0.67	85.45 ± 0.69
Qwen2.5-Coder-7B-Instruct+RLOO	86.12 ± 0.67	87.98 ± 0.89	87.01 ± 0.95	87.55 ± 0.46

The results in Table 11 demonstrate that the performances is the best when the KL constraint is moderate $\beta = 0.05$ or 0.10.

Ablation on lower-level optimization steps. The single-loop design is efficient compared to double-loop design where multiple lower-level optimization steps are performed. We include explicit training-cost comparisons to support the efficiency claim.

We vary the number of lower-level steps (1, 4, 7, 10) for RESHAPE when the base model is Qwen2.5-Coder-7B-Instruct and base RL algorithm is RLOO. We report converged performance (third row) and scaled computation time (fourth row) of reaching the corresponding performance. The computation time is scaled such that the fastest method has computation time of 1.

Table 12: Ablation on the lower-level optimization steps.

BIRD				Spider			
1	4	7	10	1	4	7	10
58.05 ± 0.66	58.12 ± 1.03	58.20 ± 0.89	58.17 ± 0.94	87.98 ± 0.89	88.22 ± 0.62	88.32 ± 0.82	88.29 ± 0.71
1.00 ± 0.06	3.61 ± 0.09	6.17 ± 0.11	9.11 ± 0.14	1.00 ± 0.02	3.54 ± 0.07	6.09 ± 0.04	8.83 ± 0.09

The results in Table 12 show that increasing the number of lower-level steps provides only marginal performance gains, while incurring substantially higher computation cost because they need to generate new samples at every lower-level step. This demonstrates that our single-loop design (1 step) achieves comparable performance to multi-step optimization, while being significantly more efficient (up to 9× cheaper).

Ablation on reward parameterization. We compare RESHAPE to the case where the shaping reward model is a linear head added on top of the final transformer layer of a language model. We consider different combinations of base models and base RL algorithms.

The results in Table 13 demonstrate that potential parameterization is better than linear reward head under different scenarios.

Table 13: Ablation on the reward parameterization

	Reward parameterization	BIRD	Spider
Qwen2.5-Coder-3B-Instruct+GRPO	linear head	50.88 ± 0.82	81.29 ± 1.07
	potential	55.49 ± 0.63	86.02 ± 0.61
deepseek-coder-6.7b-instruct+ReMax	linear head	49.24 ± 0.91	83.08 ± 1.10
	potential	54.40 ± 0.67	87.04 ± 0.62
Qwen2.5-Coder-7B-Instruct+RLOO	linear head	53.17 ± 0.55	84.88 ± 0.71
	potential	58.05 ± 0.66	87.98 ± 0.89

D Limitations

RESHAPE is limited to the scope of query generation, which is a single-turn generation. Recently, multi-turn agentic RL attracts enormous attention, and learning turn-level reward can be helpful for agentic training. It is not clear whether RESHAPE can be adapted to agentic setting and how to adapt to the agentic setting. We will explore how to adapt to agentic setting as a future work.

E Societal impact

The query generation task writes a query and uses this query to access the database/knowledge graph to retrieve information. However, a malicious user can let the model generate poisonous queries that rewrite or poison the data in the database or knowledge graph.

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 scope of this paper is restricted within query generation, which is mentioned in abstract and introduction. The contribution statement is supported by theoretical analysis and empirical 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 discuss the limitations in Appendix D.

   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 include assumptions in Assumption 1 and comoplete proof in Appendix A.

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: We include the choice of hyperparameters and training setup in Appendix B.

   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: [No]

Justification: The code is for commercial use and is not available at the time.

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 experiment details in Appendix B.

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: We report mean and standard deviation from three random seeds.

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: We include the information on computation resources in Appendix B.

   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: We follow the Code of 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: [Yes]

    Justification: We include the discussion in Appendix E.

    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: The paper poses no such risks.

    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: [Yes]

    Justification: The BIRD and Spider benchamrks are open-source, and we cite the corresponding works.

    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: The paper does not release new assets.

    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: [Yes]

    Justification: The paper uses human evaluation to check the answer correctness, which is simple as the golden answer is provided.

    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 human evaluation is conducted by the authors.

    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: The paper only uses LLM to polish the language.

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.