

Aligning LLMs Toward Multi-Turn Conversational Outcomes Using Iterative PPO

Daniel R. Jiang¹, Jalaj Bhandari¹, Yukai Yang¹, Rémi Munos², Tyler Lu¹

¹Meta, ²FAIR at Meta

Optimizing large language models (LLMs) for multi-turn conversational outcomes remains a significant challenge, especially in goal-oriented settings like AI marketing or sales agents who facilitate transactions via messaging platforms. The difficulty stems from sparse, long-horizon rewards and the discrepancy between response-level planning and token-level generation. In this technical note, we propose a formal reduction of the multi-turn RL problem into a sequence of single-turn RLHF-style problems. This is achieved by setting a learned multi-turn Q -function as the reward model for the single-turn problem. We demonstrate and prove a key insight: solving this single-turn RL problem with standard token-level PPO is equivalent to a policy improvement step within the multi-turn problem. This insight naturally leads to *Iterative PPO*, a batch online policy iteration algorithm that alternates between fitting Q -functions from logged conversation trajectories and improving the policy. A major practical advantage is that Iterative PPO directly leverages stable, off-the-shelf single-turn RLHF tools, making it straightforward to implement. Our method occupies a middle ground between fully online and fully offline approaches, retaining the adaptability of online updates while gaining the stability benefits of offline training.

Date: November 27, 2025

Correspondence: drjiang@meta.com, tylerlu@meta.com



1 Introduction

With the emergence of large language models (LLMs), there has been significant progress in building conversational AI agents (or “chatbots”) across various applications. In this work, we introduce the idea of building a conversational AI assistant that can collaborate closely with businesses to explicitly drive specific *outcomes*. The AI functions as a sales representative assistant for a business, supporting the business’s staff in engaging prospective customers. Such systems have broad potential in e-commerce, where customers interact with businesses via a sequence of *multi-turn* messages to ask questions, explore product options, negotiate prices, and ultimately make purchasing decisions. The business may seek to optimize various outcomes, such as completing a purchase, collecting customer contact information, or understanding customer requirements. In this work, we make the following contributions.

- **Outcome-driven conversational AI.** We formally define the problem of outcome-driven conversational AI that motivates this work, where an agent receives a sparse reward at the end of the conversation if an outcome is achieved. We do so through the lens of a motivating application, the *Suggested Response* problem, a variation on the widely studied chatbot problem, in which the AI agent provides response *suggestions* to the business rather than generating *automatic* replies.
- **Reduction from multi-turn to single-turn RLHF.** We show how to reduce multi-turn reinforcement learning (RL) for conversational agents at the response level to a series of single-turn RL problems at the token level. This allows us to directly leverage mature and widely available off-the-shelf single-turn RLHF tools, significantly simplifying implementation compared to prior multi-turn methods that require custom solutions. This reduction is *general* and applies to any multi-turn conversational setting with outcome-oriented goals, not just the Suggested Response problem that motivates our work.
- **Iterative PPO algorithm.** Lastly, we introduce *Iterative PPO*, a batch online policy iteration algorithm that operationalizes the *multi-turn to single-turn reduction* described above. The approach alternates

between two stages: (1) collecting a large batch of multi-turn trajectories online and (2) performing policy improvement using standard single-turn RL algorithms. This *batch online* design enables continual learning from real customer-business interactions, and importantly, does not require a simulator or model of the environment. See Figure 1 for a visualization of our approach.

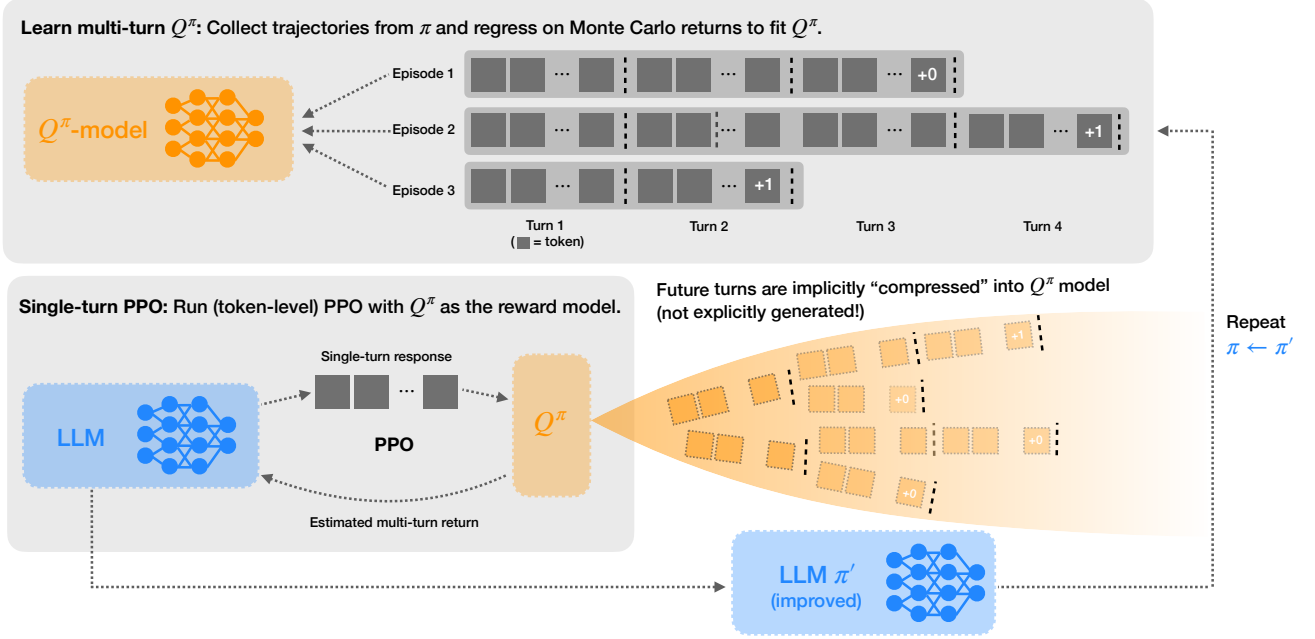


Figure 1 Reducing multi-turn to single-turn RLHF via Iterative PPO. We visualize the main steps of our proposed approach. **Top:** First, we collect multi-turn trajectories under the current policy π , compute Monte Carlo returns, and use standard reward-modeling procedures to fit Q^π , a value function that predicts the expected total multi-turn reward under π . **Bottom:** Second, holding Q^π fixed, we run standard single-turn (token-level) PPO using Q^π as the reward model. All future turns are thus implicitly “compressed” into Q^π , eliminating the need for any explicit handling of multi-turn trajectories. **Right:** Iterating this procedure ($\pi \leftarrow \pi'$) yields multi-turn improvements using only single-turn RLHF tools. Since we proceed in a series of online batches, we refer to this as “batch online.”

This paper is organized as follows. We formulate the outcome-driven problem in Section 2 and show the reduction from multi-turn to single-turn RL using our Iterative PPO approach in Section 3. We discuss related work in Section 4 and conclude in Section 5.

2 Motivating Application

In Section 2.1, we give a mathematical formulation for our motivating application, the Suggested Response (SR) problem. Then, in Section 2.2, we provide an illustrative example to provide additional intuition on the key trade-offs that exist in the SR setting.

2.1 The Suggested Response Problem

A *conversation* between a business and a customer is a sequence of *messages* $c_1, m_1, \dots, c_n, m_n$ where a customer sends message c_i and a business responds with message m_i .¹ At the end of the conversation, a binary outcome is observed.

At *turn* i of a conversation consisting of messages c_1, m_1, \dots, c_i , an AI agent assists the business by *suggesting* a message a_i for the business to send to the customer (which the business representative may or may not

¹For simplicity, consecutive messages from the same user would be concatenated to form a single message.

adopt). We call this agent the SR agent². The SR agent may also abstain from making a suggestion (e.g., if it lacks sufficient knowledge to form a relevant suggestion), denoted as $a_i = \emptyset$. For a turn i , an *outcome* o_i is an indicator variable that is true if the customer has explicitly stated their intended behavior within messages c_1, m_1, \dots, c_i , and this behavior aligns with the business’s desired actions (such as purchasing a product/service or sharing contact information). In practice, o_i can be predicted by a classifier or manually reported by the business.

There is a significant body of literature focused on the so-called *single-turn* setting: here, the focus is on selecting the next best response, rather than optimizing for long-term outcomes across multiple dialogue turns. In fact, many real-world chatbots optimize single-turn objectives (Irvine et al., 2023; Han et al., 2025).

To model the multi-turn setting, we view the conversation as a *Markov decision process (MDP)*, defined by a tuple $(\mathcal{S}, \mathcal{A}, P, P_0, R, \gamma)$ where \mathcal{S} is the state space, \mathcal{A} is the action space, P is the transition kernel, P_0 is the initial state distribution, R is the reward function, and $\gamma < 1$ is the discount factor. We discuss each of these objects in detail below.

State representation. We can represent states as the concatenation of all turns in a conversation along with the history of suggested responses and outcome indicators. That is, the initial state is $s_1 = (c_1, o_1)$ and the state at turn i is

$$s_i = (c_1, o_1, a_1, m_1, \dots, c_{i-1}, o_{i-1}, a_{i-1}, m_{i-1}, c_i, o_i), \quad (1)$$

i.e. the history until the latest customer message c_i . We use an absorbing terminal state $s_i = \perp$ to signal the end of the conversation, which occurs if any of the following hold:

1. The business’s desired outcome is achieved (i.e. $o_i = 1$), or
2. Either the business or the customer stops responding (i.e., $m_{i-1} = \emptyset$ or $c_i = \emptyset$).

Once a terminal state \perp is reached, outcomes o_i are always 0.

Actions. Given a state s_i , the action a_i is a *suggested response* by the SR agent. After the business sees the suggested response a_i , there are several possibilities of how the business’s actual message m_i is constructed: (1) the business accepts the suggested response a_i verbatim and thus $m_i = a_i$, (2) the business accepts the suggested response and then edits it, (3) the suggested response is rejected, but a message m_i is constructed with the potential influence of having seen a_i , and (4) the business outright rejects the suggested response a_i and constructs an unrelated message m_i .

Transition kernel. In state s_i , after the SR agent’s suggested response a_i , the system evolves to the next state s_{i+1} via the transition kernel $P(s_{i+1} | s_i, a_i)$. Since the terminal state is absorbing, we have $P(s_{i+1} = \perp | \perp, a_i) = 1$ (i.e., once the conversation ends, it does not restart). Embedded within $P(s_{i+1} | s_i, a_i)$, there are two steps of randomness: first, the transition to m_i (how the business actually responds after observing suggestion a_i); and second, the transition to c_{i+1}, o_{i+1} (how the customer responds to the business’s message m_i). Therefore,

$$P(s_{i+1} | s_i, a_i) = \underbrace{P(s_{i+1} | s_i, a_i, m_i)}_{\text{effect of } m_i \text{ on } (c_{i+1}, o_{i+1})} \underbrace{P(m_i | s_i, a_i)}_{\text{effect of } a_i \text{ on business's } m_i}. \quad (2)$$

The above decomposition is shown primarily to distinguish our *suggested response* model to existing works in the multi-turn interactions literature where a chatbot operates autonomously.

Our work introduces the additional consideration of $P(m_i | s_i, a_i)$, where the SR agent needs to learn how a_i impacts the business’s response m_i . Therefore, it is important that in our model, the state s_i , as defined in (1), includes both the history of suggestions (a_1, \dots, a_{i-1}) and the business’s actual messages (m_1, \dots, m_{i-1}) . By including both, we allow the agent’s state to capture the relationship between suggestions and realized messages.

Rewards. The reward only depends on the state, with $R(s_i) = 1$ if the desired outcome occurred in o_i and 0 otherwise (i.e. $R(s_i) = o_i$ where o is the most recent outcome indicator variable). Our overall goal is to find a policy $\pi(s_i)$, a distribution over actions \mathcal{A} , that maximizes the objective $\mathbb{E}[\sum_{i=1}^{\infty} \gamma^{i-1} R(s_i) | s_0 \sim P_0, a_i \sim \pi]$.

²Note the distinction between the SR agent versus an “*auto-response*” agent, which would skip the suggestion step and automatically send the reply.

2.2 Illustrative Example

To develop some intuition of our problem setting, we present an example of a hypothetical conversation in Fig. 2 and highlight the strategic aspects of planning an effective sequence of messages. Imagine an interaction between a customer who is seeking kitchen and bathroom renovation services and a home renovation business. In the left panel, the customer initiates contact to learn more by messaging the business after seeing an advertisement, mentioning that they are “thinking about updating the kitchen and maybe some bathroom work.” The SR agent now needs to suggest a message for the business to send. As we mentioned above, the business can respond to the SR agent’s suggestions in three ways: accept the message verbatim, reject the message entirely, or edit it before sending (using a “Tap to fill” button). Since conversations are open-ended and the goal is to drive outcomes, the SR agent’s role extends beyond simply answering questions: it must propose messages that are both acceptable to the business and likely to lead to an outcome. The center panel shows the SR agent’s response, along with two other responses that we include *purely to visualize suboptimal alternatives* (in reality, the LLM-based simply generates a response autoregressively as any LLM would; it does not choose between a discrete set of options).

1. The first suggestion aims to elicit more details from the customer in a generic way. It is engaging and is likely acceptable to the business, but does little to advance the conversation in a concrete way.
2. The second suggestion jumps directly to specific materials and styles. This may feel premature given the limited context on the customer specific needs. As a result, the business might reject this approach, considering it misaligned with their typical method of engaging leads.
3. The third suggestion not only asks the customer about details but proactively tries to probe deeper into their plans for kitchen renovations. In contrast to the first example, this is more outcome-driven as the business can progressively learn more about the scope of the renovation, give more details, and eventually present offerings and discounts.

We show in the right panel of Figure 2 that the business accepts the SR agent’s suggestion a_i , but makes some minor edits through the UI, resulting in m_i .

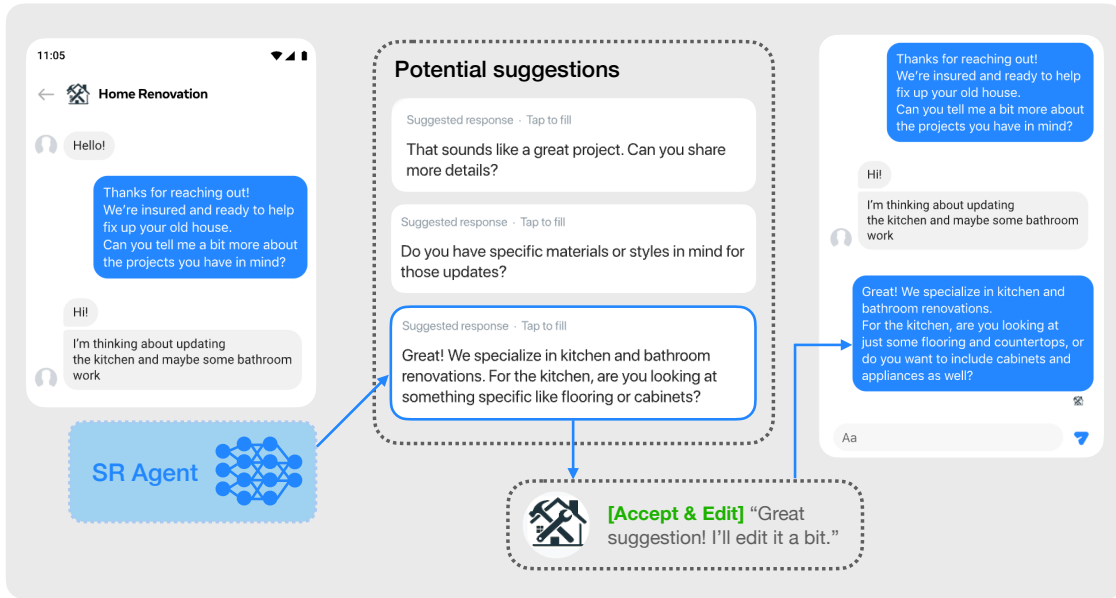


Figure 2 A hypothetical example. Left and center. Given an ongoing conversation and a customer query, the LLM-based SR agent suggests a response that trades off business acceptability and predicted downstream outcomes (note that this is shown conceptually to highlight suboptimal alternatives—in reality, the LLM agent generates a single response and does not actually select between discrete possibilities). **Right.** The business user then accepts and lightly edits the suggestion, which is sent to the customer in the chat interface.

2.3 Generalizing Beyond the Suggested Response Problem

While the Suggested Response problem above provides concrete motivation for our work, we emphasize that the technical approach we develop in the following section is not limited to this specific application. Our Iterative PPO framework applies broadly to any multi-turn conversational setting where:

- An LLM agent engages in multi-turn dialogues with specific outcome goals;
- Rewards depend on long-term conversational trajectories rather than single responses.

Specifically, our framework can handle a generic multi-turn MDP in the LLM setting, as long as states and actions are defined as sequences of tokens. The transition kernel in particular can be more general compared to (2), which encodes SR-specific logic. Indeed, our approach is applicable in many related multi-turn applications (usually involving at least one other human agent), such as general user-facing chatbots (Han et al., 2025), AI travel agents (Hong et al., 2023), customer service chatbots (Brynjolfsson et al., 2025; Ni et al., 2024), and educational tutoring agents (Shani et al., 2024; Nam et al., 2025).

3 Reducing Multi-Turn to Single-Turn RL with Iterative PPO

A key distinction between the *multi-turn* and *single-turn* RLHF settings is the following:

- In the single-turn setting, actions are typically defined at the *token level*,³ with planning performed over the sequence of tokens that constitute a response (Christiano et al., 2017; Ziegler et al., 2019). In most instantiations, the overall response is then scored according to a reward model after the end of sequence (EOS) token.
- In contrast, the multi-turn formulation naturally operates at the *response level*. This allows for conversation-level planning that occurs over the sequence of responses (Li et al., 2016; Wei et al., 2018; Shani et al., 2024).

In this work, we consider an approach inspired by dynamic programming where we reduce the initial multi-turn problem into a sequence of single-turn problems.

3.1 Iterative PPO

We propose *Iterative PPO*: at each round, we learn a Q -value function Q^π that estimates the (multi-turn) value of the current policy π at the response level, and then use single-turn, token-level RL approach (in our case PPO) where the reward function is replaced by this value function in order to generate an improved policy π' . The overall approach is called *Iterative PPO*. The following observation captures this reduction.

Observation 1 (Policy improvement via “single-turn” PPO). *Suppose that the standard single turn RLHF problem with reward function R is written $\text{SingleTurnOpt}(R)$, which is frequently solved using PPO. Now suppose that we are given the state-action value function Q^π with respect to a multi-turn policy π . Owing to the policy improvement theorem (Sutton et al., 1998), a policy π' that is greedy in the **response-level** action space with respect to Q^π has improved expected return compared to π . Since optimizing directly in the response-space is difficult, we can approximate the policy improvement step by solving the **token-level, single-turn** problem $\text{SingleTurnOpt}(Q^\pi)$. Moreover, if we use Q^* , the optimal state-action value function, as our single-turn reward, then this procedure also recovers an approximately optimal LLM policy.*

Notice that in order to guarantee policy π' improves globally over π we can relax the assumption that π' is greedy with respect to Q^π and only assume a local improvement property as expressed in the following result (whose proof is given in Appendix A).

Theorem 1. *Assume we are able to obtain a (multi-turn) policy π' which produces a single-turn improvement over π from any state s , in the sense that*

$$\mathbb{E}_{a \sim \pi'(\cdot|s)} [Q^\pi(s, a)] \geq \mathbb{E}_{a \sim \pi(\cdot|s)} [Q^\pi(s, a)].$$

³There also exists response-level actions in a single-turn setting (Ahmadian et al., 2024), but in that case the problem is only a single stage, similar to a bandit problem.

Then the resulting (multi-turn) policy π' is globally better than π , in the sense that $V^{\pi'}(s_0) \geq V^\pi(s_0)$ for any initial state s_0 .

The requirement that the new policy π' produces a single-turn improvement over π can be achieved by any RL algorithm that is effective in the setting of LLMs, such as PPO (Schulman et al., 2017), REINFORCE (Ahmadian et al., 2024), or GRPO (Shao et al., 2024), when using the values $Q^\pi(s, a)$ of the current policy as its reward values. In this work, we use PPO to implement the policy improvement step. Another version of Theorem 1, specialized to the case of KL-regularized objectives, is given in the Appendix.

Remark 1. Note that the above observation is simply one of dynamic programming or Bellman’s recursion. It is therefore not new from a classical RL perspective. Our main insight is that there exists a large body of work showing that token-level RL methods are highly effective for single-turn problems. Observation 1 essentially allows us to leverage this set of high-quality methods and their implementations.

Approximating Q^π can be achieved with standard techniques.

Observation 2 (Policy evaluation). Suppose that standard supervised reward model fitting for a dataset $\mathcal{D} = \{(x_i, y_i), v_i\}$ containing prompts x_i , responses y_i , and values v_i in the single-turn setting is written $R \approx \text{SingleTurnFitReward}(\mathcal{D})$. Consider a multi-turn policy π . In the simplest **on-policy** RL setting, we collect trajectories $\tau = (s_1, a_1, r_1, s_2, a_2, r_2, \dots, s_T)$ by rolling out π , and extract a dataset of state-action pairs and cumulative future returns $\mathcal{D}^\pi = \{(s_i, a_i), G_i\}$, where the Monte Carlo return for each time step i is

$$G_i = \sum_{j=0}^{T-i} \gamma^j r_{i+j+1},$$

with γ being the discount factor and r_{i+j+1} being the reward at time $i + j + 1$. Then, standard regression can be performed to estimate $Q^\pi \approx \text{SingleTurnFitReward}(\mathcal{D}^\pi)$.⁴

Next, notice that Observations 1 and 2 together constitute a single step of policy iteration. Hence, we see that policy iteration for the multi-turn setting can be implemented with just two single-turn methods. We call our approach *Iterative PPO*; its pseudo code is given in Algorithm 1. As we mentioned above, one of the main advantages of Iterative PPO is that it can directly leverage mature and widely available off-the-shelf single-turn RLHF tools (von Werra et al., 2020; Lefaudeux et al., 2022; Sheng et al., 2024).

We remark that our algorithm is online, but in large batches (i.e., the collection of \mathcal{D}^{π_i}). This type of *batch online* method is more stable and safer than continual online updates while being able to make larger improvements as compared to a fully offline method that is limited to optimizing within a close neighborhood of the behavioral policy.

Algorithm 1 Iterative PPO for Multi-Turn RL

- 1: Initialize π_0 (e.g. prompt an instruction-tuned model to suggest a response given conversation)
 - 2: $i \leftarrow 0$
 - 3: **repeat**
 - 4: Deploy π_i and collect trajectories $\mathcal{D}_{\text{multi}} \leftarrow \{(s_j, a_j, r_j, s_{j+1}, a_{j+1})\}$.
 - 5: $Q_i \leftarrow$ train a Q -function for π_i via Monte Carlo policy evaluation on $\mathcal{D}_{\text{multi}}$.
 - 6: Generate single-turn prompts $\mathcal{D}_{\text{single}} \leftarrow \{x_j\}$ where x_j is the prompt that π_i uses for state s_j .
 - 7: $\pi_{i+1} \leftarrow \text{PPO}(\mathcal{D}_{\text{single}}, \text{RewardModel} = Q_i)$.
 - 8: $i \leftarrow i + 1$.
 - 9: **until** Q_i has converged
 - 10: **return** π_i
-

Other methods for performing policy evaluation include SARSA and off-policy Q -evaluation. SARSA uses temporal differences (TD), which can have lower variance than Monte Carlo, but the downside is that

⁴Why do we focus on Q^π rather than aim for Q^* ? This would likely involve an algorithm like fitted Q -iteration, but note that this requires the implementation of a max operator (over responses) to generate Q -labels. This could be another PPO step, but the number of such iterations is inherently limited, since repeated policy improvement steps can quickly push the policy into regions of the response space that are out-of-distribution with respect to the original data. This is why we leverage the ability to do periodic data collection and perform Q -evaluation for the response-level problem.

TD-learning in an LLM setting is expensive and potentially unstable (Hong et al., 2024). If the data are collected using a policy other than π_i , then off-policy Q -evaluation would be required.

3.2 Practical Considerations

Implementation via a series of A/B tests. In practice, π_0 consists of simply prompting a base LLM to generate relevant suggestions based on the conversation session and other contextual information. Trajectories from π_0 can be collected by deploying it through an A/B test, from which we can learn an approximation to Q^{π_0} (see top part of Figure 1). After a policy improvement step (i.e., running PPO on Q^{π_0} ; see bottom part of Figure 1) that results in a new policy π_1 , we can deploy π_1 in an A/B test and track outcome metrics, such as the number of purchases. This A/B test for π_1 also serves to collect trajectories, from which we can repeat the above steps to arrive at π_2 , and so on. Once we are satisfied with a particular policy π_K , it can be deployed to production.

Action coverage and off-policy data collection. One important consideration is the coverage of the action space of each iteration’s collected dataset, which affects how well the learned Q -function generalizes across the action space. This can be an issue if, for example, $\pi_0(\cdot | s)$ is near deterministic and we use on-policy Monte Carlo policy evaluation. The resulting $Q^{\pi_0}(s, a)$ would therefore only be reliable at a small set of actions a that are covered by $\pi_0(\cdot | s)$. To resolve this, we may need to consider collecting data using a behavioral policy π'_0 that is significantly different from π_0 in the sense that it is much more exploratory in its actions. For example, one way to do this is via *persona exploration* where we can prompt the LLM to randomly select an array of characteristics such as tone, emoji usage, response length, etc. The trade-off with collecting data using π'_0 is that we must now perform an off-policy evaluation to learn Q^{π_0} from trajectories collected from π'_0 . However, this becomes more involved than on-policy Monte Carlo policy evaluation since it involves implementing a temporal-difference-based procedure (Lagoudakis and Parr, 2003).

4 Related Work

Our work is most closely aligned with prior works that use RL to optimize multi-turn conversation trajectories rather than single responses; that is, settings where multiple future responses are considered at each step.

Multi-turn preference optimization. Recent work on multi-turn preference optimization centers on *trajectory-level* preference feedback, where preferences are expressed over entire conversations. The MTPO algorithm of Shani et al. (2024) learns from comparisons over whole conversations and is designed for online preference collection using mirror-descent algorithm. Similar work includes Xiong et al. (2024) and Shi et al. (2024), both of which derive extensions of DPO (Rafailov et al., 2023) to the multi-turn setting. Xiong et al. (2024) proposes an iterative approach, while Shi et al. (2024) focuses on the offline setting. Our approach differs from these in a few key aspects. First, in the context of e-commerce conversational AI, collecting pairwise preference data of multi-turn trajectories is often impractical due to the length and complexity of conversations. Second, while prior works develop bespoke algorithms for multi-turn optimization, our method instead provides a simple reduction to established single-turn techniques. Lastly, although Shani et al. (2024) and Shi et al. (2024) focus on fully online and offline regimes respectively, our approach is closer to Xiong et al. (2024), operating in a hybrid setting. This enables us to retain the adaptability of online learning while mitigating the safety and stability concerns associated with continual online updates.

Hierarchical multi-turn RL. Perhaps the most closely related work is Zhou et al. (2024). Like our approach, the ArCher method distinguishes between response-level and token-level problems. However, the underlying methods differ. ArCher presents a hierarchical framework where a Q -function is fitted using off-policy Bellman updates that runs in parallel to token-level optimization using the learned Q -function as the reward model. Critically, our approach relies on the insight that the token-level optimization is in fact a policy improvement step in disguise (Thms. 1 and 2). ArCher uses methods like Implicit Q -Learning (IQL) Kostrikov et al. (2022) as the off-policy Bellman update since it does not require a direct implementation of the max operator. IQL uses an upper expectile to estimate max Q -values and it is not guaranteed to learn an optimal state-action value function.

In contrast, our reduction mechanism demonstrates that token-level optimization is a policy improvement step,

which in turn motivates our policy iteration-style algorithm. This insight, at least in principle, guarantees that our algorithm will eventually converge to an optimal policy (with enough iterations) whereas ArCHer provides no such guarantees. Another important difference lies in the simplicity of our method. ArCHer employs TD learning and optimizes multiple critic networks (Q , V , and baseline networks), which often necessitates additional tricks and extensive hyperparameter tuning to ensure stability. In contrast, our algorithm can be implemented in two straightforward phases: a supervised Q -evaluation step, followed by PPO.

POAD (Wen et al., 2024) is an orthogonal approach that extends PPO by decomposing actions into groups of tokens rather than individual tokens. REFUEL (Gao et al., 2025) simplifies ArCHer by replacing the two-step critic and policy optimization with a regression procedure. However, similar to preference optimization, REFUEL requires pairwise trajectory rollouts for each state, which can be impractical to simulate in a real-world e-commerce setting. Recently, in the context of AI tutoring, Nam et al. (2025) proposed a method that transforms the dialogue history into a compact numerical representation of the student’s state and then defines an abstract MDP over four “high-level” actions. This can be viewed as another type of hierarchical approach since the high-level action is then used to condition an LLM’s responses to the student.

Other offline approaches. Several prior works have focused on optimizing dialogue using offline methods applied to static datasets (Jaques et al., 2020; Jang et al., 2022; Snell et al., 2022; Verma et al., 2022; Chen et al., 2025). Notably, Chen et al. (2025) performs inference-time search on a model trained offline. In contrast, our approach is designed to support periodic online updates. This periodic updating not only distinguishes our method from fully offline approaches, but also enables a much simpler algorithmic framework that mirrors the evaluation and improvement steps found in classical policy iteration.

5 Concluding Remarks

We introduce a novel and real-world multi-turn RL problem where reward signals are truly dependent on the sequencing of long-term interactions. We showed how it can be reduced to a single-turn RLHF-style problem, motivating our online batch policy iteration algorithm, Iterative PPO. While our theoretical results justify our approach, we are actively experimenting with our SR agent on live traffic. Interesting avenues for future work include investigating effective exploration strategies in the message space (e.g., via personas) and determining whether our techniques have broader applicability beyond chat-based settings, such as in agentic systems.

References

- Alekh Agarwal, Nan Jiang, Sham M Kakade, and Wen Sun. Reinforcement learning: Theory and algorithms. *CS Dept., UW Seattle, Seattle, WA, USA, Tech. Rep*, 32:96, 2019.
- Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce style optimization for learning from human feedback in LLMs. *arXiv preprint arXiv:2402.14740*, 2024.
- Erik Brynjolfsson, Danielle Li, and Lindsey Raymond. Generative AI at work. *The Quarterly Journal of Economics*, 140(2):889–942, 2025.
- Zhiliang Chen, Xinyuan Niu, Chuan-Sheng Foo, and Bryan Kian Hsiang Low. Broaden your SCOPE! efficient multi-turn conversation planning for llms with semantic space. *arXiv preprint arXiv:2503.11586*, 2025.
- Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.
- Zhaolin Gao, Wenhao Zhan, Jonathan D. Chang, Gokul Swamy, Kianté Brantley, Jason D. Lee, and Wen Sun. Regressing the relative future: Efficient policy optimization for multi-turn RLHF. *arXiv preprint arXiv:2410.04612*, 2025.
- Eric Han, Jun Chen, Karthik Abinav Sankararaman, Xiaoliang Peng, Tengyu Xu, Eryk Helenowski, Kaiyan Peng, Mrinal Kumar, Sinong Wang, Han Fang, et al. Reinforcement learning from user feedback. *arXiv preprint arXiv:2505.14946*, 2025.
- Joey Hong, Sergey Levine, and Anca Dragan. Zero-shot goal-directed dialogue via rl on imagined conversations. *arXiv preprint arXiv:2311.05584*, 2023.
- Joey Hong, Anca Dragan, and Sergey Levine. Q-SFT: Q-learning for language models via supervised fine-tuning. *arXiv preprint arXiv:2411.05193*, 2024.
- Robert Irvine, Douglas Boubert, Vyas Raina, Adian Liusie, Ziyi Zhu, Vineet Mudupalli, Aliaksei Korshuk, Zongyi Liu, Fritz Cremer, Valentin Assassi, et al. Rewarding chatbots for real-world engagement with millions of users. *arXiv preprint arXiv:2303.06135*, 2023.
- Youngsoo Jang, Jongmin Lee, and Kee-Eung Kim. Gpt-critic: Offline reinforcement learning for end-to-end task-oriented dialogue systems. In *International Conference on Learning Representations*, 2022.
- Natasha Jaques, Judy Hanwen Shen, Asma Ghandeharioun, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Shane Gu, and Rosalind Picard. Human-centric dialog training via offline reinforcement learning. *arXiv preprint arXiv:2010.05848*, 2020.
- Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit Q-learning. In *International Conference on Learning Representations*, 2022.
- Michail G Lagoudakis and Ronald Parr. Least-squares policy iteration. *Journal of machine learning research*, 4(Dec): 1107–1149, 2003.
- Benjamin Lefaudeux, Francisco Massa, Diana Liskovich, Wenhan Xiong, Vittorio Caggiano, Sean Naren, Min Xu, Jieru Hu, Marta Tintore, Susan Zhang, Patrick Labatut, Daniel Haziza, Luca Wehrstedt, Jeremy Reizenstein, and Grigory Sizov. xformers: A modular and hackable transformer modelling library. <https://github.com/facebookresearch/xformers>, 2022.
- Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 1192–1202, 2016.
- Hyunji Nam, Omer Gottesman, Amy Zhang, Dean Foster, Emma Brunskill, and Lyle Ungar. Efficient RL for optimizing conversation level outcomes with an LLM-based tutor. *arXiv preprint arXiv:2507.16252*, 2025.
- Xiao Ni, Yiwei Wang, Tianjun Feng, Lauren Xiaoyuan Lu, Yitong Wang, and Congyi Zhou. Generative AI in action: Field experimental evidence on worker performance in e-commerce customer service operations. *Action: Field Experimental Evidence on Worker Performance in E-Commerce Customer Service Operations (November 06, 2024)*, 2024.

- Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.
- John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- Lior Shani, Aviv Rosenberg, Asaf Cassel, Oran Lang, Daniele Calandriello, Avital Zipori, Hila Noga, Orgad Keller, Bilal Piot, Idan Szpektor, et al. Multi-turn reinforcement learning with preference human feedback. *Advances in Neural Information Processing Systems*, 37:118953–118993, 2024.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.
- Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. HybridFlow: A flexible and efficient RLHF framework. *arXiv preprint arXiv:2409.19256*, 2024.
- Wentao Shi, Mengqi Yuan, Junkang Wu, Qifan Wang, and Fuli Feng. Direct multi-turn preference optimization for language agents. *arXiv preprint arXiv:2406.14868*, 2024.
- Charlie Snell, Ilya Kostrikov, Yi Su, Mengjiao Yang, and Sergey Levine. Offline RL for natural language generation with implicit language Q learning. *arXiv preprint arXiv:2206.11871*, 2022.
- Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.
- Siddharth Verma, Justin Fu, Mengjiao Yang, and Sergey Levine. Chai: A chatbot AI for task-oriented dialogue with offline reinforcement learning. *arXiv preprint arXiv:2204.08426*, 2022.
- Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. TRL: Transformer reinforcement learning. <https://github.com/huggingface/trl>, 2020.
- Wei Wei, Quoc Le, Andrew Dai, and Jia Li. Airdialogue: An environment for goal-oriented dialogue research. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3844–3854, 2018.
- Muning Wen, Ziyu Wan, Weinan Zhang, Jun Wang, and Ying Wen. Reinforcing language agents via policy optimization with action decomposition. *arXiv preprint arXiv:2405.15821*, 2024.
- Wei Xiong, Chengshuai Shi, Jiaming Shen, Aviv Rosenberg, Zhen Qin, Daniele Calandriello, Misha Khalman, Rishabh Joshi, Bilal Piot, Mohammad Saleh, et al. Building math agents with multi-turn iterative preference learning. *arXiv preprint arXiv:2409.02392*, 2024.
- Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. Archer: Training language model agents via hierarchical multi-turn RL. *arXiv preprint arXiv:2402.19446*, 2024.
- Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

Appendix

A Proof of Theorem 1

Proof. By the performance difference lemma (see, e.g., Section 1.5 of Agarwal et al. (2019)), we have

$$\begin{aligned} V^{\pi'}(s_0) - V^\pi(s_0) &= \frac{1}{1-\gamma} \mathbb{E}_{s \sim d_{s_0}^{\pi'}} \mathbb{E}_{a \sim \pi'(\cdot|s)} [A^\pi(s, a)] \\ &= \frac{1}{1-\gamma} \mathbb{E}_{s \sim d_{s_0}^{\pi'}} [\mathbb{E}_{a \sim \pi'(\cdot|s)} [Q^\pi(s, a)] - \mathbb{E}_{a \sim \pi(\cdot|s)} [Q^\pi(s, a)]] \geq 0, \end{aligned}$$

where

$$d_{s_0}^{\pi'}(s) = (1-\gamma) \sum_{t=0}^{\infty} \gamma^t \mathbb{P}(s_t = s | s_0, \pi') \quad (3)$$

is the discounted state visitation distribution under π' . \square

B Approximate KL-Regularized Policy Improvement

We now specialize Theorem 1 to the case of an “idealized” version of PPO (Schulman et al., 2017); namely, we include the KL-regularization term in the policy improvement objective (this is similar to the PPO objective without the clipping term)⁵. Here, we also allow for Q^π to be estimated with some maximum norm error ϵ_Q and allow the optimization objective to be optimized with some slack δ_s for each state s .

Theorem 2. *Consider a multi-turn policy π . Suppose we are given \hat{Q} , an approximation to Q^π with max-norm error $\|\hat{Q} - Q^\pi\|_\infty \leq \epsilon_Q$. Suppose we construct a new policy π' by maximizing the KL-regularized objective $\mathcal{L}(\pi') = \mathbb{E}_{s \sim P_0} \mathcal{L}_s(\pi')$, where*

$$\mathcal{L}_s(\pi') = \mathbb{E}_{a \sim \pi'(\cdot|s)} [\hat{Q}(s, a)] - \beta \text{KL}(\pi(\cdot|s) \parallel \pi'(\cdot|s)),$$

for some initial state distribution P_0 and a constant $\beta > 0$. Assume that for each s , our optimization achieves $\mathcal{L}_s(\pi') \geq \sup_\mu \mathcal{L}_s(\mu) - \delta_s$ for some error δ_s . Then, π' is an improved policy compared to π if ϵ_Q and δ_s are small enough.

Proof. By feasibility of π , we have:

$$\begin{aligned} \mathcal{L}_s(\pi') &\geq \sup_\mu \mathcal{L}_s(\mu) - \delta_s \\ &\geq \mathcal{L}_s(\pi) - \delta_s \\ &= \mathbb{E}_{a \sim \pi(\cdot|s)} [\hat{Q}(s, a)] - \delta_s \end{aligned} \quad (4)$$

where the final equality follows by the fact that the KL penalty term is 0 in $\mathcal{L}_s(\pi)$. Adding the term $\mathbb{E}_{a \sim \pi'(\cdot|s)} [Q^\pi(s, a)] - \mathbb{E}_{a \sim \pi(\cdot|s)} [Q^\pi(s, a)]$ to both sides of Equation 4 and rearranging, we get

$$\mathbb{E}_{a \sim \pi'(\cdot|s)} [Q^\pi(s, a)] - \mathbb{E}_{a \sim \pi(\cdot|s)} [Q^\pi(s, a)] \geq \beta \text{KL}(\pi(\cdot|s) \parallel \pi'(\cdot|s)) - \delta_s - 2\epsilon_Q \quad (5)$$

where we use the max-norm error bound which implies that $|\hat{Q}(s, a) - Q^\pi(s, a)| \leq \epsilon_Q$ for all (s, a) . Since $\mathbb{E}_{a \sim \pi(\cdot|s)} [Q^\pi(s, a)] = V^\pi(s)$, Equation 5 implies

$$\mathbb{E}_{a \sim \pi'(\cdot|s)} [A^\pi(s, a)] \geq \beta \text{KL}(\pi(\cdot|s) \parallel \pi'(\cdot|s)) - \delta_s - 2\epsilon_Q. \quad (6)$$

⁵Note that the policy improvement objective can be equivalently written as the “surrogate advantage objective” as introduced in TRPO (Schulman et al., 2015). To see this, $\mathbb{E}_{a \sim \pi'(\cdot|s)} [\hat{Q}(s, a)] = \mathbb{E}_{a \sim \pi(\cdot|s)} [\frac{\pi'(a|s)}{\pi(a|s)} \hat{Q}(s, a)]$.

By the performance difference lemma (see, e.g., Section 1.5 of [Agarwal et al. \(2019\)](#)), we have

$$\begin{aligned} V^{\pi'}(s_0) - V^{\pi}(s_0) &= \frac{1}{1-\gamma} \mathbb{E}_{s \sim d_{s_0}^{\pi'}} \mathbb{E}_{a \sim \pi'(\cdot|s)} [A^{\pi}(s, a)] \\ &\geq \frac{1}{1-\gamma} \mathbb{E}_{s \sim d_{s_0}^{\pi'}} [\beta \text{KL}(\pi'(\cdot|s) \parallel \pi(\cdot|s)) - \delta_s - 2\epsilon_Q], \end{aligned}$$

where $d_{s_0}^{\pi'}$ is as defined in Equation (3). Since the KL term is non-negative, for small enough δ_s and ϵ_Q , the policy π' obtained by optimizing the KL-regularized policy improvement objective is an improved policy. \square