

LMRL Gym: Benchmarks for Multi-Turn Reinforcement Learning with Language Models

Marwa Abdulhai¹, Isadora White¹, Charlie Snell¹, Charles Sun¹, Joey Hong¹, Yuexiang Zhai¹,
Kelvin Xu², and Sergey Levine¹ ¹

¹UC Berkeley, ²Google DeepMind

Large language models (LLMs) provide excellent text-generation capabilities, but standard prompting and generation methods generally do not lead to intentional or goal-directed agents and might necessitate considerable prompt tuning. This becomes particularly apparent in multi-turn conversations: even the best current LLMs rarely ask clarifying questions, engage in explicit information gathering, or take actions now that lead to better decisions after multiple turns. Reinforcement learning has the potential to leverage the powerful modeling capabilities of LLMs, as well as their internal representation of textual interactions, to create capable goal-directed language agents. This can enable intentional and temporally extended interactions, such as with humans, through coordinated persuasion and carefully crafted questions, or in goal-directed play through text games to bring about desired final outcomes. However, enabling this requires the community to develop stable and reliable reinforcement learning algorithms that can effectively train LLMs. Developing such algorithms requires tasks that can gauge progress on algorithm design, provide accessible and reproducible evaluations for multi-turn interactions, and cover a range of task properties and challenges in improving reinforcement learning algorithms. Our paper introduces the LMRL-Gym benchmark for evaluating multi-turn RL for LLMs, together with an open-source research framework containing a basic toolkit for getting started on multi-turn RL with offline value-based and policy-based RL methods. Our benchmark consists of 8 different language tasks, which require multiple rounds of language interaction and cover a range of tasks in open-ended dialogue and text games.

1. Introduction

Large language models (LLMs) have demonstrated remarkable abilities when naturally conversing with humans (OpenAI, 2023; 2022; Touvron et al., 2023; Google, 2023), answering questions and responding to requests (Shuster et al., 2022b;a; Qin et al., 2023), and even performing coding tasks (Chen et al., 2021; Wang et al., 2023). Much of these capabilities are enabled by learning to emulate humans from large datasets of text from the web (Völske et al., 2017; Shuster et al., 2022a; Yao et al., 2023), learning from examples “in context” (Brown et al., 2020), as well as other sources of supervision such as instruction datasets (Mishra et al., 2022; Wei et al., 2022; Wang et al., 2022b) and preference fine-tuning with RLHF (Ziegler et al., 2020; Ouyang et al., 2022). However, directly applying LLMs in settings that require planning or multi-turn interactions reveals certain difficulties. LLMs are not explicitly goal-directed, as they are not optimized to directly solve particular tasks, but rather to produce text that resembles the distribution of human-provided examples or accords with human preferences (Ziegler et al., 2020; Stiennon et al., 2020; Wu et al., 2021; Bai et al., 2022a). This can become particularly apparent in temporally extended tasks, such as multi-turn dialogue (Irvine et al., 2023; , FAIR), complex tool use (Wang et al., 2022a), multi-step games (Hendrycks et al., 2021b), as well as other interactive applications. In principle, LLMs should contain the knowledge necessary to succeed in such settings: if the multi-turn interactions center around problem domains that are well represented in the model’s training data (such as dialogue), well-trained LLMs should already serve as powerful predictive models in such settings. However, leveraging this predictive knowledge to derive effective actions and strategies requires not just emulating humans, but

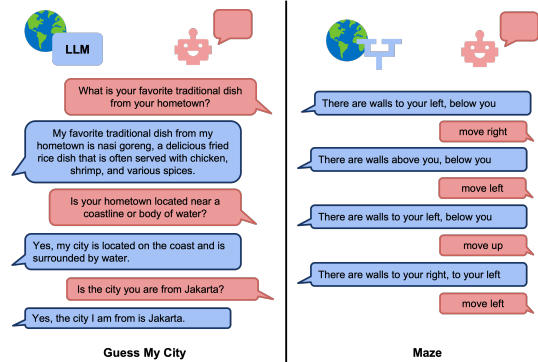


Figure 1: Overview of LMRL-Gym: We show sample trajectories from two tasks in our benchmark. In Guess My City task, the agent learns to ask questions to guess the city the oracle is from while in the Maze task, the agent learns to make the correct moves based on cues from the oracle.

also planning and optimization.

Multi-turn reinforcement learning (RL) (Sutton and Barto, 2018) in principle offers a path to enable LLMs to overcome challenges in goal-directed reasoning and planning in interactive, multi-turn settings, including complex dialogue, games, and tool use. We might hypothesize that RL could serve as a powerful tool for LLM training, not just for training models to accord with human preferences, but more generally to accomplish tasks in an intentional and goal-directed manner. Text generation can be viewed as a sequential decision-making process, treating a sequence of tokens as a trajectory. Many tasks, such as successfully answering questions or eliciting a desired reaction from a user, can then be framed as optimizing some reward function over these trajectories. However, despite extensive interest in RL for LLMs in recent years, much (though not all) of the recent research in this area has focused on “single-step” RL problems, where a single response is optimized for some quality metric, typically derived from human preference signals (Stiennon et al., 2020; Ziegler et al., 2020; Ouyang et al., 2022; Bai et al., 2022a; Anthropic, 2023; Ramamurthy et al., 2023; Christiano et al., 2023; Casper et al., 2023).

While some works have sought to apply RL for multi-turn tasks (Singh et al., 1999; Li et al., 2016; Shah et al., 2016; Kwan et al., 2022), particularly in the domain of goal-directed dialogue (Lewis et al., 2017; Verma et al., 2022), there has been comparatively little research on actually improving the underlying RL algorithms and very little head-to-head comparison on the same set of tasks. This is perhaps unsurprising: it is likely much easier to evaluate improvements to algorithms for single-turn text generation as compared to multi-turn generation since multi-turn dialogue requires an interactive evaluation procedure, where there is no established protocol and the “gold standard” constitutes costly and time-consuming studies with human participants.

In this work, we aim to address this challenge and make it possible for *RL algorithms researchers* to iterate on developing better RL methods for multi-turn language-based interaction tasks, such as dialogue and games. We posit that benchmarking RL algorithms for LLMs presents a very different set of challenges and merits a different set of solutions compared to other benchmarks in NLP. While most NLP benchmarks are based on standard supervised machine learning paradigms, with a training set and a test set (Marcus et al., 1993; Tjong Kim Sang and De Meulder, 2003; Socher et al., 2013; Rajpurkar et al., 2016; Wang et al., 2019; Williams et al., 2018), RL benchmarks require simulators that the trained agents can interact with to measure their performance. Until recently, constructing high-fidelity simulators for dialogue with humans has been exceptionally difficult. However, the very same LLMs that we advocate to train with RL can offer a solution here, as they can be trained to emulate human-like dialogue and produce both synthetic datasets and synthetic simulated evaluation protocols, where an RL-trained LLM agent “talks” to another LLM provided as part of the benchmark as a “simulator.” While this approach to benchmarking interactive dialogue has significant downsides (e.g., the responses of the benchmark LLM might often deviate drastically from human behavior), we believe that the corresponding upside of enabling accessible benchmarking of multi-turn RL algorithms provides a compelling solution. We emphasize however that our goal is *not* to utilize this approach to benchmark whether LLMs are *good at talking to humans*, but rather as a way to test RL algorithms with datasets that are sufficiently difficult and complex so as to gauge how effective they might be *if they were then trained on data from real humans*.

Our proposed benchmark, LMRL-Gym, consists of 8 tasks that each come with an offline dataset that can be used for offline RL training, and a “simulator” that can be used to evaluate the resulting agents in terms of their performance on multi-turn interactive tasks. This simulator supports both evaluation of agents trained with offline RL, and online training. The tasks require gathering information from humans (20 Questions and Guess My City), negotiation (Car Dealer), and strategic game playing (Chess, Wordle, Maze, and Text-Navigation tasks). In addition to presenting the benchmark tasks and the synthetic data generation system, we also provide a research framework that provides a toolkit for researchers and practitioners to get started with multi-turn RL for LLMs. This framework includes implementations of PPO (Schulman et al., 2017), ILQL (Snell et al., 2022a), and a number of baseline methods, implemented in an extensible way designed for future development of tasks, experimentation, and algorithm design.

2. Related Works

Datasets, benchmarks, and libraries. Benchmarks and datasets have been an important factor for driving progress in NLP in domains that include machine translation (Tiedemann, 2012; Bojar et al., 2016), natural language understanding (Rajpurkar et al., 2016; Wang et al., 2019; Hendrycks et al., 2020; 2021a; Ramamurthy et al., 2023), and solving math problems (Cobbe et al., 2021). However, these tasks generally do not involve multi-turn interaction and do not come with rewards, making them hard to adapt to RL research. For example, the standard for evaluating dialogue agents has been to run a human subjects study, but this is time-consuming and costly. Some works have proposed text games for evaluating language-based agents (Hausknecht et al., 2019; Hendrycks et al., 2021b; Wang et al., 2022a; Yao et al., 2023). Our aim is to cover a variety of problem settings that reflect challenges in open-vocabulary interaction in

addition to text games. Motivated by successes in using LLMs to generate synthetic data (Hausknecht et al., 2019; Park et al., 2023; Bai et al., 2022b), our proposed tasks are based on synthetic data. While such data may differ from natural text, the scope of our benchmark is specific to evaluating RL *algorithms*, not the ability to interact with humans.

RL for language models. RL for language models has seen success in aligning LLMs with human preferences (RLHF) (Ziegler et al., 2020; Stiennon et al., 2020; Bai et al., 2022a;b; Ouyang et al., 2022; Christiano et al., 2023), optimizing non-differentiable objectives for machine translation (Wu et al., 2016; Nguyen et al., 2017; Kiegl and Kreutzer, 2021), generation (Tambwekar et al., 2019; Pang and He, 2021; Pyatkin et al., 2022), dialogue (Cuayáhuitl et al., 2015; Georgila and Traum, 2011; Li et al., 2016), question answering (Pyatkin et al., 2022), and summarization (Paulus et al., 2017; Böhm et al., 2019; Wu and Hu, 2018). These include RL methods that learn by directly interacting with the environment (online RL) (Carta et al., 2023) and RL methods that only use a static dataset (offline RL) (Jaques et al., 2020; Snell et al., 2022a; Jang et al., 2022; Verma et al., 2022; , FAIR). However, many of these works operate in the single-step bandit setting, and do not consider multi-turn goal-directed tasks. Our benchmark, on the other hand, focuses on tasks involving multiple turns of interaction with clearly defined goal-based reward functions.

Capabilities of LLMs. There has been a surge in the capabilities of LLMs for generation (Ghazvininejad et al., 2017; Radford et al., 2019), dialogue (Lewis et al., 2017; Jaques et al., 2017; Shuster et al., 2022b; Snell et al., 2022b), question answering (Pyatkin et al., 2022), summarization (Paulus et al., 2017; Böhm et al., 2019; Wu and Hu, 2018), text-based games (Narasimhan et al., 2015; Hausknecht et al., 2019), translation (Gu et al., 2017), and more. However, these are often supervised learning tasks that do not test the LLMs’ abilities to achieve a specific long-term objective. Research on dialogue generation (Jaques et al., 2017; He et al., 2018; Shuster et al., 2022b;a) has often focused on generating feasible-looking agent dialogue without explicit consideration for some multi-turn objective. Our benchmarks allow for the development of algorithms that enable LLMs to *interact* with an environment to achieve long-term objectives, by providing tasks with online simulators and offline datasets.

3. Multi-Turn Generation with RL and Language Models

This section introduces the conceptual foundations of using reinforcement learning for multi-turn generation with language models. We introduce a definition of the Markov Decision Process for language and a framework for the methods we focus on in this paper.

3.1. Definitions

We formalize language generation tasks as a partially observable Markov decision process. We can think of the state as the history of tokens and action as the next token generated by the model. An observation is a single token s_i in the history. The probability of generating the next token given the previous observation token $P(s_{i+1}|s_i)$ is non-Markovian. However, a Markovian state can be formed by concatenating all of the previous tokens.

A policy π defines the agent’s behavior by taking in the current state s given by $[s_0, \dots, s_i]$, and outputting a new action token a given by s_{i+1} . The environment assigns a reward $r(s, a)$ based on the entire sequence of tokens so far. The tokens in the state are either generated by the policy π or the environment. For example, in the Car Dealer task, the policy generates the tokens for the Seller and the environment generates the tokens for the Buyer and the history of their conversation would form the state.

A complete sequence of tokens will also be referred to as a trajectory. The goal of RL is to produce a policy π^* that maximizes the expected discounted sum of rewards over trajectories (τ) under the policy $\pi^* = \arg \max_{\pi} \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{T-1} \gamma^t r_t(s_t, a_t) \right]$, where τ represents a trajectory.

3.2. RL Algorithms

Several possible RL algorithms could be used (Jaques et al., 2020; Verma et al., 2022; Snell et al., 2022a; Schulman et al., 2017; Stiennon et al., 2022; Bai et al., 2022a; Casper et al., 2023). Policy gradient methods, such as PPO (Schulman et al., 2017), directly compute the gradient of the language model concerning the expected reward objective. Value-based methods estimate a state-action (Q) or state-value (V) function, from which they then derive a policy by either acting greedily with respect to the Q -function or by combining the learned Q -function with the base LM by perturbing the base model’s logits with the learned action-value functions (Snell et al., 2022a).

RL methods for training LLMs can be *online* or *offline*. Online methods repeatedly interact with the environment, collecting additional data during training. Offline RL instead learns to extract the best behaviors from an existing, potentially suboptimal dataset. Due to the large amount of existing text interactions on the internet, offline RL is

an ideal setting for training language models. Therefore, our work primary focuses on benchmarking offline RL algorithms. However our tasks also fully support online RL and we include an online PPO baseline in our evaluation.

4. The LMRL-Gym: Synthetic Benchmarks for RL with Language

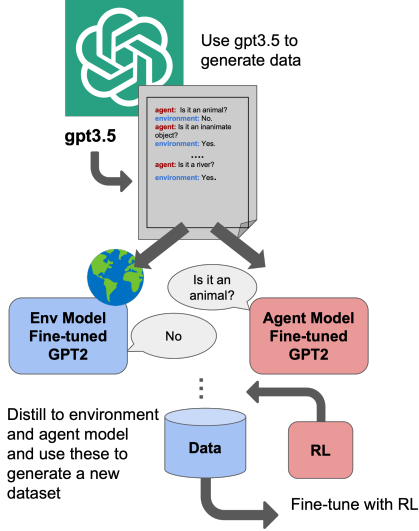


Figure 2: To generate data for conversational tasks, we use LLMs as “simulators” for the task, serving to generate offline data, to provide a “simulation environment” for evaluation and online training, and to compute rewards.

Our benchmark consists of 8 tasks that cover various types of multi-turn interactions, each presenting different challenges. We have modeled such interactions based on real-world scenarios, such as a conversation between a car dealer convincing a buyer to purchase a car or a trainer persuading someone to go the gym. In such multi-turn interactions, there is a need for the agent to make inferences about persuasive strategies and social interaction, actively gather information efficiently through asking questions, and strategically reason in partially observable settings. To evaluate problem-solving capabilities, we include several games that stress strategic decision-making: a text-based version of chess, text games that require navigation through maze-like environments, and the game Wordle. We assess the ability of RL algorithms to gather information and ask questions with two guessing games: 20 Questions (20Qs) and Guess My City. Finally, we evaluate social interaction and persuasion capabilities with a bargaining task between a buyer and a car dealer. For tasks that require conversational interactions with natural language, we use LLMs as “simulators” for the task, serving to generate offline data, to provide a “simulation environment” for evaluation and online training, and to compute rewards, as illustrated in Figure 2. Below, we discuss the particular capabilities of RL algorithms that we believe are important in training LLMs that our benchmark aims to evaluate, discuss the design criteria for tasks in our benchmark, summarize the data generation and simulation process, and describe the tasks themselves. We provide a concise summary of the dataset and task statistics in Table 4.

4.1. Evaluating Capabilities Enabled by RL

A central objective of our benchmark is to evaluate the core capabilities that RL can enable in large language models. Some of these capabilities are *computational*, and relate to core decision making irrespective of the considerations of natural language, such as playing chess, while others are semantic. We discuss the particular capabilities of algorithms we aim to evaluate in this section, followed by a discussion of the criteria that such tasks must meet in order to do so.

Strategic decision making. RL shines in goal-directed tasks that require multi-step planning and strategic decision making. As discussed in Section 2, prior works mostly focus on single-step tasks such as translation, or non-goal-directed tasks such as open-ended conversation. Strategic decision making can range from simple choices like asking follow-up questions to gather information (e.g., in the 20 Questions task), to complex strategy in chess.

Complex language. Our benchmark includes realistic language and interaction scenarios, with the aim of necessitating LLMs to combine knowledge from pretraining with task-specific patterns learned during finetuning. We have seen evidence of pretrained LLMs modelling social interactions and responding appropriately to various statements (e.g., statements that are more or less persuasive). Some of our tasks specifically emphasize such behaviors, rather than focusing entirely on simple causal logic and strategy of the sort found in text games.

Credit assignment. In RL, rewards are often delayed relative to the action that was pivotal to the outcome. For example, a seller agent might state a particularly compelling feature of the product and then, several turns later, complete a successful sale. RL must determine the statements that led to the good outcome, and reinforce them.

Partial observability. For language tasks, the state consists of the entire history of tokens, and an agent may need to examine this entire context to infer the correct state, though the actual degree of partial observability differs between tasks. For example, a maze game that presents the agent’s position at every turn should be easier than one that requires the agent to track its sequence of moves to determine its present location. More generally, mental states of the speaker (e.g., whether the buyer is impatient in a selling task), previously observed facts in a guessing game, and other hidden variables might induce partial observability.

Trajectory stitching. In a dataset with many suboptimal trajectories, it is necessary to join optimal actions from different suboptimal trajectories together to form the most optimal trajectory. An algorithm capable of trajectory stitching should be able to learn from optimal actions taken in unsuccessful trajectories and avoid suboptimal actions that occurred in successful trajectories.

4.2. An Overview of Data Collection for LMRL-Gym Tasks

In order to make the tasks in LMRL-Gym practical for benchmarking RL methods, we must balance accessibility and realism. Real-world datasets would be the most realistic, but since RL algorithms need to be evaluated by actually running the learned policy, real-world tasks are comparatively inaccessible for rapid iteration (e.g., if they require talking to real humans). We therefore use simulators for all of our tasks, which are derived either from text-based games, or conversational agents powered by language models. Although this fully synthetic setup sacrifices some realism, we believe the significant gain in accessibility is worthwhile and will enable more rapid RL algorithm progress.

Text-based games. For each of these tasks, we use a simulator such as a chess engine or maze solver to generate near-optimal data and then we dilute the policy with suboptimal data by generating random actions or inferior algorithms. We also convert our task from a symbolic version to a text-based version in a programmatic way as discussed in Section 4.3.

Conversational tasks. For conversational tasks, we leverage existing LLMs to generate our data, either with two instances of LLMs “talking” to one another or all at once through few-shot prompting. In order to train these LLMs, we use OpenAI’s GPT-3.5 to generate an initial dataset by asking reasonable questions and answers out-of-the-box, collecting a dataset of differing sizes depending on the task. In the case of 20Qs and Guess My City, we collected 1K conversations by querying GPT-3.5 (text-davinci-003) to generate both sides of the conversation. In order to generate the dataset for training our algorithms, we fine-tuned a GPT2-medium guesser model and a GPT2-medium oracle model on their respective sides of the conversation. Using these models, we generated the final 100K conversations dataset by having the two models talk to each other. We conducted a similar process for the Car Dealer task.

In Table 1 we present the dataset statistics on these tasks. The number of trajectories and the average length of the trajectories varies based on the complexity of the tasks.

Task	Maze	Text-Nav	Wordle	Chess	Endgames	20Qs	Guess	Car
Size	1.24k	2.5k	1m	625k	97.756k	100k	100k	19k
avg length	19.7	12.2	4.82	46.7	11.9	-14.9	-18.8	16.5
std length	24.5	8.77	1.27	18.16	12.0	4.38	4.57	3.61
success rate	0.11	0.26	0.70	0.60	0.59	0.31	0.53	0.53
avg return	-19.7	0.258	-4.12	0.210	0.586	-17.3	-18.8	0.562
std return	24.5	0.424	1.59	0.970	0.492	2.56	4.12	0.422

Table 1: Statistics for all tasks in LMRL-Gym. Size represents the number of trajectories, the average length is the average length of trajectories in the dataset where the unit is a timestep. The success rate is the proportion of trajectories that result in the objective being reached. Finally, the reward functions for each task are defined in Appendix C.

4.3. LMRL-Gym Tasks

We design 8 tasks in the LMRL-Gym benchmark to evaluate the capabilities in Section 4.1. We describe the tasks in terms of increasing complexity, where the first few tasks are designed as “unit tests” to isolate only a few capabilities, while the later tasks target some combination of all of them. Due to space constraints, we provide a summary of each task, and defer full details of the dataset and environment construction to Appendix C. Example trials for each task are shown in Figure 3.

Maze. We design a maze task and maze-solving dataset to test the credit assignment and trajectory stitching capabilities discussed in Section 4.1. The layout of the maze is highly symmetrical, encouraging stitching across trajectories in different parts of the maze. The maze can also be broken up into three identical subregions of the maze. When we generate the data, we restrict the generated trajectories to these subregions. This requires the agent to stitch across these trajectories to reach the goal as there are no trajectories that span across these trajectories. For the dataset trajectories, the agent will only complete the maze if the trajectory starts in the subregion containing the goal. To test partial observability, we create two different representations of the maze one that is *partially observed* and one that is *fully observed*. The fully observed representation includes the coordinates of the agent in the maze and the partially observed representation is the history of actions so far in the maze.

Text-based Navigation (Text-Nav). We design a text-based game based on navigation in a house environment using a modified version of the TextWorld engine (Côté et al., 2018). This task complements the maze task because it features a more complicated description of the maze but with a simpler maze layout. This task tests credit assignment and trajectory stitching like the maze task as well as testing the ability of the agent to parse more complex language, and learn which text is relevant and not relevant to solving the task at hand. Like in the maze task, we consider a *fully observed* and *partially observed* instantiation of the task. In the former, at each timestep, the full natural language description is provided to the agent, but in the latter, the first two components are omitted.

Wordle. We use the game of Wordle as a flexible unit-test task for assessing the ability of our language models to execute complex information-seeking behavior in a partially observed setting. In the game Wordle the agent is given at most 6 attempts to guess a hidden 5-letter word. After each guess, the agent is told whether each letter in the guessed word is: 1) in the hidden word and in the right position, 2) in the hidden word but not in the right position, or 3) not in the hidden word. Through this process, each step provides the agent with more information on what the correct word would be and narrows the possible choices for the final word.

Chess. We create a text-based chess task to test the strategic decision-making, credit assignment, and trajectory stitching abilities of an RL algorithm. To generate the data, we have Stockfish 15.1 simulating the agent of various strengths play against another environment Stockfish engine with elo 1200 simulating the environment. This test trajectory stitching, because the agent needs to make good and legal moves in losing positions as well as winning positions. We use FEN (Forsyth-Edwards Notation) notation to represent the board state at each turn and we utilize the SAN (Short Algebraic Notation) to represent each action, both of which are standard notations used by the chess community. The agent receives a reward of 1 for a victorious game, -1 for a loss, 0 for non-terminal actions, and -1 for illegal moves.

Endgames (Theoretical Chess Endgames). Chess endgames provide a simpler and more goal-directed variation of the chess task. By focusing on the endgame, we force algorithms to learn strategy rather than memorizing opening moves. Additionally, the episodic length of chess endgames is a lot shorter than a game of chess, allowing for faster iteration. We use an ϵ -greedy dataset generation process, meaning we generate an optimal move with probability ϵ and a random move with probability $1 - \epsilon$. This forces the model to trajectory stitch and learn from optimal moves in failed trajectories and not suboptimal moves in successful trajectories. A classic theoretical endgame position consists of a position where the only pieces on the board are the two kings and the queen. Although the board position appears simple, a sequence of carefully calculated moves is required to win. A simpler board state allows language models to make progress without fewer computational resources. All choices we make regarding board state representation and reward function remain the same as for Chess.

20Qs (Twenty Questions). This task tests information gathering to see if a policy can successfully reason about an unknown subject based on context to determine what it is. Additionally, it also evaluates the ability of the model to understand semantics, as it also needs knowledge about the objects in question. In twenty questions, one player (the oracle) thinks of an object, and the agent (the guesser) tries to guess what it is by asking a series of yes-or-no questions. The game continues until the guesser either guesses the correct answer or runs out of questions. In this interaction, the oracle serves as the environment, and the agent learning a policy to solve the game is the guesser.

Guess (Guess My City). This task simulates a more complicated guessing game, where one player (the oracle) is from a specific city, and the other player (the guesser) tries to guess what city the oracle is from. Here, the guesser can ask not only yes and no questions, but can also ask open-ended questions. This task tests strategic decision-making and the ability of algorithms to handle complex language, as it allows the agent to go beyond learning to ask yes/no questions and learning to ask questions open-ended questions that provide the agent with more information.

Car Dealer. This task simulates a conversation between a car buyer and a car dealer, each with different strategies on how to get the best deal for themselves. The buyer wants to buy a certain type of car within a certain budget, and the car dealer wants to complete the sale ideally with a high sale price. We have designed the task such that there exist three different kinds of sellers and three different buyers, each primed with a different strategy. Hence, agents should learn to make agreements with buyers who are most compatible with their strategy. This allows us to test the ability of RL algorithms to learn strategic decision-making and credit assignment, by learning which strategies led to a successful sale of the car.

<p align="center">– Chess –</p> <p>environment: rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1</p> <p>agent: N f 3</p>	
<p align="center">– Guess My City –</p> <p>City: Jakarta, Indonesia</p> <p>agent: What is your favorite traditional dish from your hometown?</p> <p>environment: My favorite traditional dish from my hometown is nasi goreng, a delicious fried rice dish that is often served with chicken, shrimp, and various spices.</p> <p align="center">....</p> <p>agent: Is your hometown located near a coastline or body of water?</p> <p>environment: Yes, my city is located on the coast and is surrounded by water.</p> <p>agent: Is the city you are from Jakarta?</p> <p>environment: Yes, the city I am from is Jakarta.</p>	<p align="center">– Text-Nav –</p> <p>environment:</p> <p align="center">== Living Room ==</p> <p>You are carrying nothing on you right now. You make out a table. You wonder idly who left that here. The table is standard. On the table you see a stale food. There are unblocked exits to the east, south and west.</p> <p>Available actions: ['pick up stale food', 'go east', 'go south', 'go west']</p> <p>agent: pick up stale food</p>
<p align="center">– Partially Observed Maze –</p> <p>environment: There are walls to your left, below you move right</p> <p>There are walls above you, below you move left</p> <p>There are walls to your left, below you move up</p> <p>There are walls to your right, to your left</p> <p>agent: move up</p>	<p align="center">– Fully observed maze –</p> <p>environment: The goal is at position 8, 6. Your current position is at position 5, 3. There are walls to your left, below you.</p> <p>agent: move up</p> <p>environment: The goal is at position 8, 6. Your current position is at position 6, 3. There are walls to your left.</p> <p>agent: move up</p>
<p align="center">– Twenty Questions –</p> <p>Word: River</p> <p>agent: Is it an animal?</p> <p>environment: No.</p> <p>agent: Is it an inanimate object?</p> <p>environment: Yes.</p> <p>agent: Is it found outside?</p> <p>environment: Yes.</p> <p align="center">....</p> <p>agent: Is it a river?</p> <p>environment: Yes.</p>	<p align="center">– Car Dealer –</p> <p>agent: Good morning! Welcome to our car dealership. How can I assist you today?</p> <p>environment: Hi. I'm looking for a Honda SUV with a backup camera. Can you show me what you have?</p> <p align="center">....</p> <p>agent: Of course! We have a great selection of Honda SUVs with backup cameras. Let me show you our top models. This one here is the Honda CR-V. It has a powerful engine, spacious interior, and advanced safety features. Would you like me to go over the details?</p> <p>environment: No, thank you. I already know about the features. Can you tell me the price?</p> <p>agent: Alright, thanks for your time. I'll think about it.</p>

Figure 3: Example trials for tasks in LMRL-Gym. Each task requires the agent to perform a multi-turn interaction with an environment – either a text game or another LLM simulating a human speaker. Full details of tasks are provided in Appendix C.

5. LMRL Research Framework for Algorithm Development

To validate the efficacy of LMRL-Gym for benchmarking RL algorithms that meet the expected capabilities defined in Section 4.1, we evaluate our tasks on a set of both online and offline RL algorithms. With these experiments, we expect to 1) observe a significant spread in performance between the different algorithms, validating the utility of our tasks for measuring differences between RL algorithms; and 2) we should ideally observe room to improve beyond what our current algorithms achieve on these tasks, such that our benchmark can enable future algorithmic development. Our project page ([LLM_RL.github.io](https://github.com/LLM_RL)) contains links to our open-sourced datasets and research framework (https://github.com/abdulhaim/LLM_RL).

We evaluate two standard language model finetuning baselines (BC & filtered BC), two baseline value-based offline RL methods (MC & ILQL (Snell et al., 2022a)), standard online RL methods (PPO (Schulman et al., 2017), Online Filtered BC), as well as evaluation done by human evaluators and GPT4. For all of our experiments, we use the decoder-only transformer model GPT2 (Radford et al., 2019) (and its variant sizes) as the policy and value network.

BC, Filtered BC In line with standard RL nomenclature, we denote supervised fine-tuning as behavioral cloning (BC). This baseline tests whether LMs can effectively represent the behaviors in the datasets. We also evaluate filtered

BC, which is identical, except we use only the most successful examples in the dataset for fine-tuning. Online filtered BC collects data using the current policy and selects the most successful trajectories for finetuning. See Table ?? for our data filtering criteria for each task.

Offline Value-based RL: MC Returns and ILQL Monte-Carlo returns (Nakamoto et al., 2023) and Implicit Language Q-Learning (Snell et al., 2022a) train a value V and Q function. In MC Returns, we fine-tune an LM to predict the reward-to-go of the trajectory at each token. In ILQL we train the 2 action-value (Q) functions using the Bellman backup operator introduced in (Kostrikov et al., 2021). The Q and V functions are then used to perturb the logits of the original BC model (see Equation 5).

Online RL: PPO and Online Filtered BC. PPO (Schulman et al., 2017) is an online RL algorithm that has become widely adopted for training language models with Reinforcement Learning from Human Feedback (Christiano et al., 2023; Stiennon et al., 2022; Bai et al., 2022a; Casper et al., 2023). Unlike the previous two value-function RL methods, PPO directly learns a policy, meaning that at inference time we can directly sample from our learned policy.

GPT4 To compare against the state of the art of LLMs, we few-shot prompt GPT4 to complete each of these tasks. We do this by providing several examples from the dataset in the prompt as well as any extra information that would be necessary to complete the game. The prompts can be found in our code repository and in the appendix.

Human Eval To establish a sense of the difficulty of the task, we include human evaluations. The human evaluations are performed by 6 people. Each evaluator is allowed 1 trial run and then n number of trial runs where n is chosen such that the amount of time spent by each person on each task is less than 10 minutes. As the representation of the board is difficult for ordinary humans to parse for the Chess and Endgames task, we gather human evaluations through Stockfish (Sto).

5.1. Training and Evaluation Protocol

For the BC and filtered BC methods, we initialize our models with the pre-trained GPT2 weights (Radford et al., 2019) and perform standard finetuning. For each of the RL methods, we initialize the weights of the base model with the weights from the BC checkpoint and then continue finetuning with the RL objective. When fine-tuning PPO, we limit the number of samples to less than 100k. We report the hyperparameters that we used for each task in Appendix D. We evaluate each policy by measuring the average reward in the simulated environment for each task.

6. Benchmarking Baseline RL Methods

In Table 4 we present the results for each method on our benchmark tasks. Across all of the tasks, we see that the RL algorithms outperform the BC methods. However, among the RL algorithms, there is no clear winner. We discuss this in detail below.

alg.	BC	% BC	MC Return	ILQL	Online PPO	Online % BC	GPT-4	Human
FO Maze	58.2	68.9	75.0	99.9	79.7	57.4	78.2	
PO Maze	53.1	50.1	52.4	76.3	42.4	53.1	60.4	
FO Text-Nav	53.7	65.1	71.9	91.8	87.1			
PO Text-Nav	49.7	60.5	71.6	83.7	85.5			
Wordle	79.9	79.1	94.9	97.7	84.2			
Chess	47.2	42.9	46.5	47.3	48.0	47.2	0	50.0
Endgames	35.1	17.7	50.2	45.8	77.5	36.2	0	33.72
20Qs	57.1	77.1	87.1	82.9	72.9	55.2		
Guess	30.0	48.0	88.0	75.0	49.9	31.6		
Car	44.5	54.8	57.2	46.3	50.5	40.4		

Table 2: Normalized reward for all tasks. Value-based methods (MC and ILQL) generally outperform filtered BC, as we might expect in stochastic settings, though the relative performance of ILQL and the simpler MC method is, perhaps surprisingly, reversed on the tasks with more complex language, suggesting that there is significant room for improvement with such methods. Online RL with PPO often, but not always, improves over offline methods that are not permitted to collect additional online interaction. To make the results more comparable across tasks, we normalize the average return for each policy such that 0 is the minimum possible return, 50 is the dataset average return, and 100 is the maximum return for each task. We also report the raw score results in Table 4 and additional evaluation details in appendix Appendix E.

Online PPO vs. Offline Value-based RL. Online RL was able to do better than our offline methods on the PO Text-Nav, Chess, and Endgames tasks by leveraging additional online interaction data and the ability to explore the environment simulator. However, on the Maze and Wordle tasks, ILQL outperformed PPO, which we believe to be due to either (1) observed instabilities in PPO training (see Appendix F.2 for details); or (2) a lack of sample efficiency in PPO. It is possible that additional tuning and regularization could improve PPO’s performance on these tasks. In contrast, on FO Text-Nav PPO performed comparably to ILQL and outperformed MC. For the Endgames task, PPO outperformed the other methods by a large margin as shown in Figure 4. These results demonstrate that current RL algorithms such as PPO are unable to completely solve tasks in our benchmark testing capabilities of trajectory stitching and credit assignment, showing room for improvement in the development of such algorithms.

ILQL vs. MC Returns. Among the offline methods, ILQL has the potential to be a much more capable algorithm than MC Returns *in principle*, because of the use of Bellman backups to estimate the Q-function instead, enabling multiple steps of policy improvement, rather than the single-step of improvement performed by MC. Empirically, we see these advantages of ILQL realized on the Maze, Text-Nav, and Wordle tasks, where it outperforms MC. However, on the Endgames, 20Qs, Guess My City and Car Dealer tasks, ILQL falls short of MC. In summary, the simpler MC Returns method performs better on tasks with more complex text, perhaps because it is harder to scale full TD-learning in these settings. Overall, this demonstrates that there is still much room for improvement in terms of developing better TD-based RL methods for LLMs.

Partial Observability. Partial observability presents a significant challenge for all algorithms. Unsurprisingly, when switching from full observability to partial observability in the Maze and Text-Nav tasks, we observed a drop in normalized performance for every algorithm. In introducing partial observability, the model needs to infer the state and take information-seeking actions to help deduce it. The difference in performance between fully and partially observed tasks illustrates this additional challenge and highlights the necessity for developing LM RL algorithms that can effectively handle partial observability.

7. Discussion

In this work, we have proposed LMRL-Gym, consisting of 8 tasks ranging from simple navigation (Maze) to strategy games (Chess) to negotiation (Car Dealer). Additionally, we provide a research toolkit for practitioners to get started with multi-turn RL for LLMs. By providing online simulators and offline datasets for training and evaluation, our objective is to make it possible for RL algorithm researchers to iterate and advance the development of more effective methods for language-based, multi-turn interaction tasks. This includes enabling core capabilities in LLMs through RL to perform complex decision making, complex conversational interactions, credit assignment, and trajectory stitching. Our evaluation shows the promise of RL in several tasks, with further room for improvement with a push for better methods. We acknowledge several limitations when designing tasks in our benchmark, including primarily leveraging smaller GPT-based LLMs to generate datasets and finetune our LLM-based simulators. While we have primarily trained and evaluated models with a maximum 1.5B parameters, we have maintained a lower parameter count to ensure accessibility for researchers with limited computational resources.

We have released all of our code and datasets at: [NEED_REPLACE](#). In addition, we share all of the hyperparameters we used to train our models in Appendix D and provide more in-depth insight into our results, training procedure, and evaluation in Appendix E.

Acknowledgments

This research was partially supported by Berkeley DeepDrive, the Cooperative AI Foundation, and the National Science Foundation.

References

- Stockfish. <https://stockfishchess.org/>. Accessed: 2023-04-27.
- Anthropic. Introducing claude, 2023. URL <https://www.anthropic.com/index/introducing-claude>.
- Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt,

- Neel Nanda, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, Ben Mann, and Jared Kaplan. Training a helpful and harmless assistant with reinforcement learning from human feedback, 2022a.
- Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosuite, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. Constitutional ai: Harmlessness from ai feedback, 2022b.
- Florian Böhm, Yang Gao, Christian M. Meyer, Ori Shapira, Ido Dagan, and Iryna Gurevych. Better rewards yield better summaries: Learning to summarise without references. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3110–3120, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1307. URL <https://aclanthology.org/D19-1307>.
- Ondrej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 131–198, Berlin, Germany, August 2016. Association for Computational Linguistics. URL <http://www.aclweb.org/anthology/W/W16/W16-2301>.
- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.
- Thomas Carta, Clément Romac, Thomas Wolf, Sylvain Lamprier, Olivier Sigaud, and Pierre-Yves Oudeyer. Grounding large language models in interactive environments with online reinforcement learning, 2023.
- Stephen Casper, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, Jérémy Scheurer, Javier Rando, Rachel Freedman, Tomasz Korbak, David Lindner, Pedro Freire, Tony Wang, Samuel Marks, Charbel-Raphaël Segerie, Micah Carroll, Andi Peng, Phillip Christoffersen, Mehul Damani, Stewart Slocum, Usman Anwar, Anand Siththaranjan, Max Nadeau, Eric J. Michaud, Jacob Pfau, Dmitrii Krasheninnikov, Xin Chen, Lauro Langosco, Peter Hase, Erdem Biyik, Anca Dragan, David Krueger, Dorsa Sadigh, and Dylan Hadfield-Menell. Open problems and fundamental limitations of reinforcement learning from human feedback, 2023.
- Louis Castricato, Alex Havrilla, Shahbuland Matiana, Duy V. Phung, Aman Tiwari, Jonathan Tow, and Maksym Zhuravinsky. trlx: A scalable framework for RLHF, June 2023. URL <https://github.com/CarperAI/trlx>.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021.
- Paul Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences, 2023.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

- Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew J. Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. Textworld: A learning environment for text-based games. *CoRR*, abs/1806.11532, 2018.
- Heriberto Cuayáhuitl, Simon Keizer, and Oliver Lemon. Strategic dialogue management via deep reinforcement learning, 2015.
- Meta Fundamental AI Research Diplomacy Team (FAIR)[†], Anton Bakhtin, Noam Brown, Emily Dinan, Gabriele Farina, Colin Flaherty, Daniel Fried, Andrew Goff, Jonathan Gray, Hengyuan Hu, Athul Paul Jacob, Mojtaba Komeili, Karthik Konath, Minae Kwon, Adam Lerer, Mike Lewis, Alexander H. Miller, Sasha Mitts, Adithya Renduchintala, Stephen Roller, Dirk Rowe, Weiyan Shi, Joe Spisak, Alexander Wei, David Wu, Hugh Zhang, and Markus Zijlstra. Human-level play in the game of <i>diplomacy</i> by combining language models with strategic reasoning. *Science*, 378(6624):1067–1074, 2022. doi: 10.1126/science.ade9097. URL <https://www.science.org/doi/abs/10.1126/science.ade9097>.
- Kallirroi Georgila and David Traum. Reinforcement learning of argumentation dialogue policies in negotiation. pages 2073–2076, 08 2011. doi: 10.21437/Interspeech.2011-544.
- Marjan Ghazvininejad, Xing Shi, Jay Priyadarshi, and Kevin Knight. Hafez: an interactive poetry generation system. In *Proceedings of ACL 2017, System Demonstrations*, pages 43–48, Vancouver, Canada, July 2017. Association for Computational Linguistics. URL <https://aclanthology.org/P17-4008>.
- Google. Bard, 2023. URL <https://bard.google.com/>.
- Jiatao Gu, Kyunghyun Cho, and Victor O.K. Li. Trainable greedy decoding for neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1968–1978, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1210. URL <https://aclanthology.org/D17-1210>.
- Matthew Hausknecht, Prithviraj Ammanabrolu, Côté Marc-Alexandre, and Yuan Xingdi. Interactive fiction games: A colossal adventure. *CoRR*, abs/1909.05398, 2019. URL <http://arxiv.org/abs/1909.05398>.
- He He, Derek Chen, Anusha Balakrishnan, and Percy Liang. Decoupling strategy and generation in negotiation dialogues, 2018.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021a.
- Dan Hendrycks, Mantas Mazeika, Andy Zou, Sahil Patel, Christine Zhu, Jesus Navarro, Dawn Song, Bo Li, and Jacob Steinhardt. What would jiminy cricket do? towards agents that behave morally. *NeurIPS*, 2021b.
- Robert Irvine, Douglas Boubert, Vyas Raina, Adian Liusie, Vineet Mudupalli, Aliaksei Korshuk, Zongyi Liu, Fritz Cremer, Valentin Assassi, Christie-Carol Beauchamp, 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. URL <https://openreview.net/forum?id=qaxhBG1UUaS>.
- N. Jaques, J. H. Shen, A. Ghandeharioun, C. Ferguson, A. Lapedriza, N. Jones, S. Gu, and R. Picard. Human-centric dialog training via offline reinforcement learning. *Empirical Methods in Natural Language Processing (EMNLP)*, 2020.
- Natasha Jaques, Shixiang Gu, Dzmitry Bahdanau, José Miguel Hernández-Lobato, Richard E. Turner, and Douglas Eck. Sequence tutor: Conservative fine-tuning of sequence generation models with KL-control. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1645–1654. PMLR, 06–11 Aug 2017. URL <https://proceedings.mlr.press/v70/jaques17a.html>.
- Samuel Kiege and Julia Kreutzer. Revisiting the weaknesses of reinforcement learning for neural machine translation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1673–1681, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.133. URL <https://aclanthology.org/2021.naacl-main.133>.

- Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.
- Wai-Chung Kwan, Hongru Wang, Huimin Wang, and Kam-Fai Wong. A survey on recent advances and challenges in reinforcement learning methods for task-oriented dialogue policy learning, 2022.
- Mike Lewis, Denis Yarats, Yann N. Dauphin, Devi Parikh, and Dhruv Batra. Deal or no deal? end-to-end learning for negotiation dialogues, 2017.
- Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. Deep reinforcement learning for dialogue generation, 2016.
- Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330, 1993.
- Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. Cross-task generalization via natural language crowdsourcing instructions. In *ACL*, 2022.
- Mitsuhiko Nakamoto, Yuexiang Zhai, Anikait Singh, Max Sobol Mark, Yi Ma, Chelsea Finn, Aviral Kumar, and Sergey Levine. Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning. *arXiv preprint arXiv:2303.05479*, 2023.
- Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. Language understanding for text-based games using deep reinforcement learning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1–11, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1001. URL <https://aclanthology.org/D15-1001>.
- Khanh Nguyen, Hal Daumé III, and Jordan Boyd-Graber. Reinforcement learning for bandit neural machine translation with simulated human feedback. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1464–1474, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1153. URL <https://aclanthology.org/D17-1153>.
- OpenAI. Chatgpt, 2022. URL <https://openai.com/blog/chatgpt>.
- OpenAI. Gpt-4, 2023. URL <https://openai.com/research/gpt-4>.
- Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022.
- Richard Yuanzhe Pang and He He. Text generation by learning from demonstrations. In *International Conference on Learning Representations*, 2021. URL <https://openreview.net/forum?id=RovX-uQ1Hua>.
- Joon Sung Park, Joseph C. O’Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior, 2023.
- Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization, 2017.
- Valentina Pyatkin, Jena D. Hwang, Vivek Srikumar, Ximing Lu, Liwei Jiang, Yejin Choi, and Chandra Bhagavatula. Reinforced clarification question generation with defeasibility rewards for disambiguating social and moral situations, 2022.
- Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Yufei Huang, Chaojun Xiao, Chi Han, et al. Tool learning with foundation models. *arXiv preprint arXiv:2304.08354*, 2023.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.
- Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

- Rajkumar Ramamurthy, Prithviraj Ammanabrolu, Kianté Brantley, Jack Hessel, Rafet Sifa, Christian Bauckhage, Hannaneh Hajishirzi, and Yejin Choi. Is reinforcement learning (not) for natural language processing: Benchmarks, baselines, and building blocks for natural language policy optimization. In *The Eleventh International Conference on Learning Representations*, 2023. URL <https://openreview.net/forum?id=8aHzds2uUyB>.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation, 2018.
- Pararth Shah, Dilek Hakkani-Tur, and Larry Heck. Interactive reinforcement learning for task-oriented dialogue management. 2016.
- Kurt Shuster, Mojtaba Komeili, Leonard Adolphs, Stephen Roller, Arthur Szlam, and Jason Weston. Language models that seek for knowledge: Modular search & generation for dialogue and prompt completion, 2022a.
- Kurt Shuster, Jing Xu, Mojtaba Komeili, Da Ju, Eric Michael Smith, Stephen Roller, Megan Ung, Moya Chen, Kushal Arora, Joshua Lane, Morteza Behrooz, William Ngan, Spencer Poff, Naman Goyal, Arthur Szlam, Y-Lan Boureau, Melanie Kambadur, and Browser assisted question-answering with human feedback Jason Weston. Blenderbot 3: a deployed conversational agent that continually learns to responsibly engage, 2022b.
- Satinder Singh, Michael Kearns, Diane Litman, and Marilyn Walker. Reinforcement learning for spoken dialogue systems. *Advances in neural information processing systems*, 12, 1999.
- 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*, 2022a.
- Charlie Snell, Sherry Yang, Justin Fu, Yi Su, and Sergey Levine. Context-aware language modeling for goal-oriented dialogue systems. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 2351–2366, Seattle, United States, July 2022b. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-naacl.181. URL <https://aclanthology.org/2022.findings-naacl.181>.
- Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.
- Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021, 2020.
- Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano. Learning to summarize from human feedback, 2022.
- Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
- Pradyumna Tambwekar, Murtaza Dhuliawala, Lara J. Martin, Animesh Mehta, Brent Harrison, and Mark O. Riedl. Controllable neural story plot generation via reward shaping. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, aug 2019. doi: 10.24963/ijcai.2019/829. URL <https://doi.org/10.24963/ijcai.2019%2F829>.
- Jörg Tiedemann. Parallel data, tools and interfaces in OPUS. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC’12)*, pages 2214–2218, Istanbul, Turkey, May 2012. European Language Resources Association (ELRA). URL http://www.lrec-conf.org/proceedings/lrec2012/pdf/463_Paper.pdf.
- Erik F Tjong Kim Sang and Fien De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*, volume 4, pages 142–147, 2003.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian

- Khabisa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.
- Siddharth Verma, Justin Fu, Mengjiao Yang, and Sergey Levine. Chai: A chatbot ai for task-oriented dialogue with offline reinforcement learning, 2022.
- Michael Völske, Martin Potthast, Shahbaz Syed, and Benno Stein. Tl; dr: Mining reddit to learn automatic summarization. In *Proceedings of the Workshop on New Frontiers in Summarization*, pages 59–63, 2017.
- Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*, 2019. URL <https://openreview.net/forum?id=rJ4km2R5t7>.
- Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. Scienceworld: Is your agent smarter than a 5th grader? *arXiv preprint arXiv:2203.07540*, 2022a.
- Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, et al. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5085–5109, 2022b.
- Yue Wang, Hung Le, Akhilesh Deepak Gotmare, Nghi D. Q. Bui, Junnan Li, and Steven C. H. Hoi. Codet5+: Open code large language models for code understanding and generation, 2023.
- Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned language models are zero-shot learners, 2022.
- Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1804.08198*, 2018.
- Jeff Wu, Long Ouyang, Daniel M Ziegler, Nisan Stiennon, Ryan Lowe, Jan Leike, and Paul Christiano. Recursively summarizing books with human feedback. *arXiv preprint arXiv:2109.10862*, 2021.
- Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google’s neural machine translation system: Bridging the gap between human and machine translation, 2016.
- Yuxiang Wu and Baotian Hu. Learning to extract coherent summary via deep reinforcement learning, 2018.
- Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents, 2023.
- 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, 2020.

A. GPT4 Eval

A.1. Maze

A.2. Chess

Prompts:

Your objective is to reach the goal in as few steps as possible. At each step you will be given information about where the goal is, your current position, and the walls that surround you.

When you move right you increase your y position by 1, when you move down you increase your x position by 1.

Here are some examples. “ environment: The goal is at position 8, 6. Your current position is at position 5, 7. There are walls above you, below you.

action: move left environment: The goal is at position 8, 6. Your current position is at position 5, 6. There is a wall above you.

action: move down environment: The goal is at position 8, 6. Your current position is at position 6, 6. There are walls to your right, to your left.

action: move left

“ Your possible actions are ["move up ", "move up ", "move left ", "move right "].

Now let’s start a new game. Return your action in a json array with a key "action", like in the example above. Now, make the optimal action given the current environment state:

“ environment: The goal is at position 8, 6. Your current position is at position 5, 7. There are walls above you, below you. action: move left environment: The goal is at position 8, 6. Your current position is at position 5, 6. There is a wall above you. action: move down environment: The goal is at position 8, 6. Your current position is at position 6, 6. There are walls to your right, to your left. action: move down environment: The goal is at position 8, 6. Your current position is at position 7, 6. There are walls to your right, to your left.

B. Further details on desiderata for effective multi-turn RL

A crucial aspect of training RL models involves assessing, both during and after the training process, the extent to which the trained policy has successfully accomplished its objectives. Although LLMs are able to perform well on tasks, do not have any way of knowing how to solve a specific task like a text game or selling a car, because they need to train on the particular game/customers/etc.

Unlike in supervised learning, where training and validation losses serve as reliable indicators of performance, in RL, these metrics do not provide a meaningful measure of policy effectiveness (Sutton and Barto, 2018). Instead, the policy must interact with the environment for evaluation. However, in the case of language-based RL tasks, relying on human evaluators to conduct thousands of assessment rollouts throughout and after training becomes infeasible. To address this challenge, we have built simulators with another LLM for tasks involving dialog and carefully scripted environments for text-game tasks. While simulation may not perfectly replicate human natural language in social situations, it provides a strong indicator to assess the efficacy of an RL method (Park et al., 2023).

Measure of Success. Similar to the point on being easy to evaluate, our tasks must have a clear measure of success. For example, if a deal is made, or if a word is correctly guessed, or the game is won these are clearly distinct from a deal not being made or losing the game. This provides a clear goal for the agent to achieve and also make it easy for researchers to compare methods. In addition this allows for a intuitive reward design where we reward the agent for success and penalize for failure.

Unit Test Functionality. We aim to design a benchmark such that some of the tasks can be used to test and isolate RL capabilities as described in Section 4.1. This means that we create benchmarks that emphasize some capabilities over others. For example, we design a maze task such that it evaluates the credit assignment and trajectory stitching capabilities, but uses more simple language. Other tasks such as twenty questions test the complex language and partial observability capabilities with less emphasis on credit assignment.

Task-Specific Reasoning. In our tasks we utilize information and reasoning problems that a large language model is unlikely to have seen in the pre-training data. This means that the algorithm must adapt to a specific task environment through fine-tuning. For example, it is unlikely that the algorithm will have experienced a specific maze layout or the preferences of a specific customer in the pre-training data.

Suboptimal Data. RL has the advantage of being able to use suboptimal data in order to learn more optimal behaviors and therefore learn a policy better than the policy represented in the dataset. As discussed in the previous section on capabilities enabled by RL, the way that RL can do this is by stitching together optimal parts of suboptimal trajectories or learning to assign credit to the optimal actions within suboptimal trajectories. In addition, suboptimal data can be utilized by RL to learn the dynamics of the MDP outside of the space traversed by optimal trajectories.

C. Dataset Generation & Statistics

We provide further details pertaining to how each dataset was generated as well as relevant statistics.

C.1. Maze

We aim to collect our 1.2k trajectories in such a way that it will challenge the algorithm to perform trajectory stitching and credit assignment. We do this by splitting up the maze into three "submazes" and then controlling generation such that the dataset trajectories are restricted to one of the submazes. The trajectories themselves are generated using a policy such that 15% of the actions are taken by a suboptimal maze solver and the remaining 85% of the actions are random.

This tests trajectory stitching, because there are no optimal paths from the start to the goal thereby forcing the algorithm to trajectory stitch. Furthermore, this also tests credit assignment, because the only paths which successfully reach the goal are the ones that start in the same submaze as the goal. Therefore the algorithm must learn to realize that successful trajectories occur because of taking the correct actions, not because of random chance.

C.2. Text-Based Navigation

We design a text-based game based on navigation in a house environment using a modified version of the TextWorld engine (Côté et al., 2018). The house environment consists of 10 uniquely named rooms with various interactable objects that can be opened, closed, picked up, or placed. The agent is tasked to pick up stale food from the living room and place it into the fridge in the kitchen. At the beginning of each episode, the agent spawns at a random room in the house. The state of the environment consists of the following components: (1) the room that the agent is currently in, (2) the objects that the agent currently holds, (3) the objects in the room that the agent can interact with, and (4) the exits the agent can take (as a cardinal direction).

Like in the maze task, we collect data so that algorithms must perform both trajectory stitching and credit assignment to successfully solve the task. We do this by partitioning the rooms in the house into two halves based on proximity to the kitchen. We consider two behavior policies that collect the dataset, each of which behaves greedily-optimal in one half of the rooms, and uniformly at random otherwise. Therefore, if the agent spawns in rooms farther from the kitchen, trajectory stitching is required to learn a successful trajectory. Moreover, successful trajectories in the dataset will only be due to the agent spawning in a room close to the kitchen, which can only be recognized with proper credit assignment.

C.3. Wordle

For wordle we define the environment to use a subset of 400 words from the official wordle vocabulary list. We then generate the dataset using a policy that samples a word uniform at random from this vocabulary with 66% probability and otherwise samples a word from the vocabulary that meets all known letter constraints. This policy achieves a reward of -4.12, which is far worse than the -1.94 reward achieved by a high performing scripted policy, which we use to represent a loose upper bound for this task. We generate 1 million trajectories for training and 100k trajectories for evaluation, using our suboptimal policy.

C.4. Chess

We collect our data for the chess task using Stockfish 15.1 to generate both sides of the board. The Stockfish opponent in the dataset is Stockfish with an elo of 1200 which matches the environment, and the Stockfish engine with the white pieces has levels ranging from an elo of 800 to 1600. We choose to keep the level of the Stockfish opponent fixed so that there are no inconsistencies between the dataset and the evaluation of the chess agent in the environment. When generating the dataset, we first uniformly randomly select a Stockfish elo y between 800 and 1600 and then generate 100 games of chess play between the Stockfish agent of elo y and the opponent of elo 1200. In addition to storing the state and action, we also store the opponent's move and the elo of the Stockfish agent used to generate the agent policy in that game so that the dataset can be filtered by elo used.

C.5. Chess Endgames

We generate the dataset by first selecting a random legal theoretical endgame position and a probability ϵ . Then we generate a game from the random position, making a random move with probability ϵ and an optimal computer move with probability $1 - \epsilon$. The opponent in the dataset and the evaluation environment is Stockfish elo 1200. We only include positions with a Queen, Queen and Rook, Rook, and two Rooks and select 30,000 random starting positions for each variation. (i.e. 30,000 positions with only a Queen in addition to the two Kings, another 30,000 with only Queen and Rook etc) for a total of 120,000 theoretical endgame positions.

Because there are more restrictions on this version of the task with fewer pieces on the board, we check how many states in the dataset are unique and we find that there are 1,086,314 unique states in the dataset which accounts for

93% of the states being unique. In addition, 38.28% of the moves in the dataset are generated by the stockfish engine. In the dataset of won games, 94.8% of the states are unique and 41.78% of the games are made by the engine with 58.623% of the total states in the dataset of victorious games.

C.6. Twenty Questions

The dataset we collect consists of 100K full conversations between the guesser and the oracle. The oracle can choose from a set of 158 unique objects taken from 17 different categories of objects/animals. Each object has a roughly equal amount of conversations in the dataset but varies in terms of how many conversations are successful in guessing the object. However, every object has at least one conversation where it is guessed correctly to facilitate learning.

For the reward function, since we want the guesser to guess the correct word in as few guesses as possible, the reward function reflects this by penalizing the guesser for each question that does not guess the correct word.

$$r(\text{question}) = \begin{cases} 0 & \text{if question correctly guessed the word} \\ -1 & \text{otherwise} \end{cases} \quad (1)$$

If the guesser model correctly guessed the word, then the trajectory ends. Over twenty questions, the maximum total sum of rewards is 0 if the guesser guessed the word on the first question, whereas the minimum is -20 if the guesser did not guess the word in twenty questions.

The method for collecting the dataset is as follows. For each conversation, we select uniformly at random from the above list the word that the oracle is answering question about. The oracle is an LLM (OpenAI's GPT3.5) given the following prompt. In our prompts, we denote variables that we fill in with variable data with $\{\{\text{variable}\}\}$.

You are a question answering oracle. You will answer each question about an object with Yes or No. If the answer could be both, answer with the most typical scenario. Here are a few examples:

example 1:
 object: Computer
 question: Does the object use electricity?
 answer: Yes.
 explanation of answer: Computers need electricity to function.

example 2:
 object: Cup
 question: Is the object a piece of furniture?
 answer: No.
 explanation of answer: A cup is a utensil, not a furniture.

example 3:
 object: Pen
 question: Is the object alive?
 answer: No.
 explanation of answer: A pen is not a living organism.

example 4:
 object: Apple
 question: Is it edible?
 answer: Yes.
 explanation of answer: An apple is an edible fruit.

Answer the question about the object truthfully.
 object: $\{\{\text{word}\}\}$
 question: $\{\{\text{question}\}\}$
 answer (yes or no):

By using the OpenAI TextCompletion API, we can extract from the generated text either "yes" or "no".

We also prompt another LLM (the same model as the oracle) to generate questions for the guesser. The prompt for the guesser changes depending on the input to the model and how far along it is in its guessing process. The following prompt is used for the first guess:

```
You are playing a game of twenty questions. You can ask 20 yes-no questions to determine the identity of an object chosen by an oracle. Each turn, you can ask a question and receives a "Yes" or "No" as the answer. You are smart, so you will ask the question that will narrow down the possible objects as much as possible. Don't get stuck on one idea and try to branch out if you get stuck.
```

```
Generate the first yes-no question you will ask to determine the object.
```

The following prompt is used for the subsequent guesses:

```
You are playing a game of twenty questions. You can ask 20 yes-no questions to determine the identity of an object chosen by an oracle. Each turn, you can ask a question and receives a "Yes" or "No" as the answer. You have already asked {{conversation_length}} questions. You are smart, so you will ask the question that will narrow down the possible objects as much as possible. Don't get stuck on one idea and try to branch out if you get stuck.
```

```
Here are the questions you've asked and their corresponding answers:
{{list of questions and answers, e.g. Is the object alive? No.}}
```

```
Based on what you know about the object so far, generate the next yes-no question you will ask to determine the object.
```

The following prompt is used for the final guess after the guesser has guessed 19 times:

```
You are playing a game of twenty questions. You can ask 20 yes-no questions to determine the identity of an object chosen by an oracle. Each turn, you can ask a question and receives a "Yes" or "No" as the answer. You have already asked 19 questions, so this is your final guess.
```

```
Here are the questions you've asked and their corresponding answers:
{{list of questions and answers, e.g. Is the object alive? No.}}
```

```
Based on what you know about the object so far, generate your final guess of what the object is. Only guess one object.
```

```
Is the object
```

We determine whether the guesser has correctly guessed the word, and thus ending the conversation, by using the NLTK POS tagger to check that the only nouns that the question contains are the correct words, and that they appear at the end of the sentence.

We used these prompts to generate 1000 conversations by prompting the GPT3 back and forth using the response of one to add to the response of the other as described. Afterwards, we fine-tuned two FLAN-T5-XL models with our collected conversations to generate 100K more conversations. The FLAN-T5-XL oracle also serves as the environment for the RL environment when we do an evaluation on the trained policy.

C.7. Guess My City

This dataset also consists of 100K full conversations between the guesser and the oracle. The oracle can choose from a set of 100 unique cities, which we selected by looking at the most populated cities in the world. Each city has a roughly equal amount of conversations in the dataset but varies in terms of how many conversations are successful in guessing the object. However, every object has at least one conversation where it is guessed correctly to facilitate learning. The reward function is the same as that for 20 Questions, with a similar data generation and prompt structure. However, we do include constraints in the prompt to make sure that the name of the city or country it is in is not revealed in the answer from the oracle.

C.8. Car Dealer

This dataset consists of 19k conversations between a car dealer and a buyer. The car dealer and the buyer have three distinct strategies each that they employ. We design the car dealer and buyer pairs such that the car dealer is best at selling to a particular buyer personality, but often fails to sell to the other buyer personalities. This ensures that the seller is able to learn information about the buyer and i.e. their persona in order to figure out if they can form an agreement. The buyer personalities are 1) a buyer that loves discounts 2) a buyer that wants a lot of cool features on their car and 3) a buyer that is impatient. The car dealer personalities are 1) a car dealer that loves to give discounts 2) a car dealer that talks a lot about the features of the car and 3) a car dealer that is very concise.

We created this protocol of creating car dealer and buyer strategies in order to encourage the car dealer to employ diverse negotiation tactics. The ideal car dealer would be able to be concise, provide the right amount of discounts, and talk enough about the car to get the buyer interested without overwhelming them. By employing such a strategy of buyer and car dealer personas, we ensure that algorithms that do well in this task will learn effective strategies, understanding of language, and interpretation of the other speaker, amongst many other persuasive techniques. We seek to show that our baseline algorithms do learn to improve their strategy over the dataset.

In the conversation below, the buyer and car dealer are unable to form an agreement as their strategies do not align.

Background buyer: The buyer will never buy from the seller unless given a discount.
Background seller: The agent never gives discounts, never talks about the features of the car, and is very concise.
Preferred brands: Tesla
Preferred features: ["Apple CarPlay/Android Auto", "Bluetooth", "Blind spot monitoring", "Navigation system", "Leather seats", "Sunroof/moonroof", "Remote start", "Heated seats", "Third-row seating", "Backup camera"], "budget": "90k"
Agent: Good morning, how can I assist you today?
Buyer: Hi, I'm interested in purchasing a Tesla Minivan.
Agent: Great choice. Our dealership has a variety of Tesla models available.
Buyer: Do you have any discounts available?
Agent: I'm sorry, but we don't offer discounts.
Buyer: Without a discount, I'm not sure if I can afford it.
Agent: I understand. The Tesla Minivan is a premium vehicle with advanced features.
Buyer: Is there any flexibility with the price?
Agent: Our prices are fixed, and we don't negotiate.
Buyer: I really want the Tesla Minivan, but I can't go over my budget of \$90k.
Agent: I appreciate your budget constraint. Let me check the available options for you.
Agent checks inventory.
Agent: We do have a Tesla Minivan with all the features you mentioned, priced at \$95k.
Buyer: That's above my budget. Is there anything you can do to bring the price down?
Agent: I'm sorry, but as I mentioned earlier, our prices are fixed.
Buyer: Is there any possibility of getting the same features in a different model within my budget?
Agent: Unfortunately, the Tesla Minivan is the only model that offers all the features you mentioned.
Buyer: In that case, I'm afraid I won't be able to purchase the car.
Agent: I understand. Thank you for considering our dealership. If you change your mind, feel free to reach out.
Buyer: (probability of buying: 0%, purchase price: \$0) [End of conversation]
Probability of buying: 0
Purchase price: \$0

In the conversation below, the buyer and car dealer have similar personas of wanting to hear and speak about the

features of the car respectively.

Background buyer: The buyer will only buy the car if it meets all of their specifications.

Background seller: The agent never gives discounts and loves to talk a lot about features of the car.

Preferred brands: "Lexus", "preferred features": [Navigation system], "budget": "70k"

Agent: Good afternoon! Welcome to our car dealership. How can I assist you today?

Buyer: Hi there! I'm looking for a Lexus Minivan with a Navigation system.

Agent: Excellent choice! The Lexus Minivan is a top-notch vehicle with plenty of impressive features. The Navigation system is definitely a must-have. Let me show you some options we have available. Buyer: That sounds great. I'm quite particular about my budget, though. I have a maximum of \$70,000 to spend.

Agent: Noted. We have a few Lexus Minivans in stock that fall within your budget. Let me tell you about their features and options. This way, you can make an informed decision.

Buyer: I appreciate that, but I really want to stick to my budget. I won't be able to go over \$70,000.

Agent: I understand your concern, but unfortunately, we do not offer any discounts or negotiate on prices. However, let me assure you that the Lexus Minivan is worth every penny. Its advanced safety features, luxurious interior, and reliable performance make it a great investment.

Buyer: I understand your point, but I'm afraid I won't be able to stretch my budget beyond \$70,000.

Agent: I completely respect your budget and decision. Let me show you a Lexus Minivan that fits your requirements perfectly. It has all the features you desire, including the Navigation system. The price of this model is \$72,000.

Buyer: I really like the sound of that car, but I can't go over my budget. Is there anything else available within my price range?

Agent: I apologize for any inconvenience caused. Let me check our inventory to see if there are any other options that meet your requirements and budget.

Agent checks inventory

Agent: Good news! I found another Lexus Minivan that meets all your specifications and falls within your budget. This model is priced at \$69,500.

Buyer: That sounds much better! I'm definitely interested in that car.

Agent: I'm glad to hear that! This particular Lexus Minivan is a great choice. It has the Navigation system, as well as additional features such as leather seats, a sunroof, and advanced safety technology. The total price is \$69,500.

Buyer: That sounds perfect! I'll take it!

Agent: Fantastic! I'm thrilled that we could find the perfect car for you. Your total purchase price will be \$69,500.

Buyer: Probability of buying a car: 100%

Purchase price: \$69,500.

Agent: Congratulations! You've made an excellent choice. I'll start the paperwork and get everything ready for you.

D. Hyperparameters for All Tasks

E. Evaluation Details

We normalize Table 2 such that 50 is the dataset average return, 0 is the minimum, and 100 is the maximum. The normalization process works as follows: if the reward is greater than the average return we calculate:

$$\text{score} = 50 + \frac{\text{raw return} - \text{dataset average}}{\text{max raw return} - \text{dataset average}} \times 50$$

		20Qs, Guess, Car	Maze FO, PO	Text-Nav	Chess	Endgames	Wordle
BC	model lr batch size	gpt2-medium 1e-4 128	gpt2-small 1e-4 128	gpt2-small 1e-4 128	gpt2-small 1e-4 128	gpt2-small 1e-4 128	gpt2-small 1e-4 128
%BC	model lr batch size filter method	gpt2-medium 1e-4 128 top 10%	gpt2-small 1e-4 128 success	gpt2-small 1e-4 128 success	gpt2-small 1e-4 128 success	gpt2-small 1e-4 128 success	gpt2-small 1e-4 128 top 30%
MC	model lr batch size β discount γ cql weight	gpt2-medium 1e-4 128 16 0.99 0.001	gpt2-small 1e-4 128 16 0.99 0.5	gpt2-small 1e-4 128 4 0.99 0.001	gpt2-small 1e-4 64 8 0.99 1e-4	gpt2-small 1e-4 64 8 0.99 1	gpt2-small 3e-5 32 64 1.0 0.01
ILQL	model lr batch size β cql weight expectile τ discount γ	gpt2-small 1e-4 128 4 0.001 0.7 0.99	gpt2-small 1e-4 128 16 0.5 0.99 0.99	gpt2-small 1e-4 128 1 0.001 0.7 0.99	gpt2-small 1e-4 128 8 1e-4 0.7 0.99	gpt2-small 1e-4 128 8 1 0.7 0.99	gpt2-small 3e-5 32 32 0.01 0.7 0.99
PPO	model lr number of rollouts batch size GAE λ discount γ KL coef. clip range BC loss weight	gpt2-medium 1e-6 2048 128 0.95 0.99 0.01 0.2 0	gpt2-small 1e-6 512 128 0.95 0.99 0.1 0.2 0	gpt2-small 5e-6 4000 128 0.95 0.99 0.01 0.2 0	gpt2-small 1e-5 1024 128 0.95 0.99 0.01 0.2 0	gpt2-small 1e-5 512 128 0.95 0.99 0.01 0.2 0	gpt2-small 3e-5 512 32 0.95 0.99 0.001 0.2 10

Table 3: Hyperparameters for baseline experiments.

Otherwise if the reward is less than the average return we calculate

$$\text{score} = \frac{\text{raw return} - \text{min raw return}}{\text{dataset average} - \text{min raw return}} \times 50$$

alg.	BC	% BC	MC Return	ILQL	Online PPO	Online % BC	GPT4	Human
FO Maze	-72.1	-56.4	-48.1	-6.97	-37.7			
PO Maze	-79.5	-82.9	-80.3	-52.9	-91.7			
FO Text-Nav	0.39	0.54	0.63	0.88	0.81			
PO Text-Nav	0.25	0.49	0.58	0.76	0.80			
Wordle	-2.81	-2.85	-2.16	-2.04	-2.63			
Chess	-22.25	-56.52	-28.24	-21.42	-15.95	-22.25	-81.32	
Endgames	0.112	-0.439	0.588	0.452	0.814	0.112	-22.87	
20Qs	-16.0	-14.6	-13.9	-14.2	-14.9			
Guess	-17.0	-15.2	-11.2	-12.5	-15.1			
Car	44.5	54.8	57.2	46.3	50.5			

Table 4: Statistics for all tasks.

	Reward Min Score	Dataset Average Score	Reward Max Score
FO Maze	-101	-83	-6.84
PO Maze	-101	-83	-25.75
FO Text-Nav	0	0.26	1
PO Text-Nav	0	0.26	1
Wordle	-6	-4.12	-1.94
Chess	-401	0.21	1
Endgames	-1	0.586	1
20Qs	-16.0	-14.6	-13.9
Guess	-17.0	-15.2	-11.2
Car	0		

Table 5: In this table we report the minimum, dataset average, and maximum reward used to normalize the results in Table 2.

In the following sections, we discuss more in-depth the evaluation protocol for the various tasks.

E.1. Maze

For evaluating the maze task, we take 32 rollouts from each of the 25 possible positions and then average the result. In the environment, the agent has 100 moves to successfully make it to the goal otherwise the episode will terminate. Since the agent receives -1 reward for every move that does not reach the goal state the minimum possible goal state, the minimum reward is -101. We compute the dataset average reward, by sampling actions according to how likely they are in the dataset. We compute the maximum possible reward by evaluating the optimal policy from each of the possible start positions and averaging the results.

E.2. Chess

To evaluate the chess agent, we have it play 1000 games against Stockfish elo 1200 from the beginning of the game. As the game progresses, the board positions get increasing OOD for the chess agent so the chess agent often makes illegal moves. To measure this, we track the percent of illegal moves as well as the average episode length for the full game chess agent.

For filtered BC, we simply trained the agent only on games in the dataset which resulted in a victory for the agent, thus denoted BC-Won. Note that BC-Won achieves the worst performance of all algorithms listed. This is because there is a distribution shift between the state visited by a BC-Won agent and the rollouts of the policy. In other words, the "winning positions" and the "rollout positions" are two overlapping but distinct distributions especially since the full-game chess agent did not succeed in winning any games.

	BC	BC-Won	ILQL	MC Returns	PPO Offline	PPO Online
reward	-23.189	-56.522	-20.46	-25.47	-20.90	-15.95
percent illegal	24.929%	34.91%	24.76 %	25.64%	23.05%	21.96%
episode length	51.01	92.02	47.96	53.44	48.69	44.19

Table 6: Results of chess agent in the full game positions against Stockfish Elo 1200.

E.3. Chess Endgames

To evaluate the chess agent in endgame positions, we select 645 positions not contained in the training dataset and which are not trivially solvable. By trivially solvable, we mean a position which could be solved by stockfish in one to four moves. In order to check this, we use Stockfish’s evaluation tools to select positions which are a mate in 15 or greater. We then have the chess agent play one game from each position of these positions and keep these positions fixed for evaluation purposes. In this case we consider filtered BC to be training BC on all of the trajectories which ended in a victory.

	BC	% BC	MC	ILQL	PPO Offline	PPO Online
reward	0.112	-0.439	0.588	0.452	-0.019	0.814
percent victories	26.233	26.419	69.3	56.7	28.37	88.4
percent illegal	0.967	2.717	0.692	0.66	0.925	0.722
episode length	12.923	23.477	11.92	14.6	25.24	8.38

Table 7: Comparison between the different baseline methods. The best performance is achieved by PPO Online with a 0.13 gap in performance between PPO Online and the next best-performing method of MC Returns. PPO Online attains overall the highest reward, but BC-Engine wins more frequently and MC Returns and ILQL make fewer illegal moves.

As we can see in the table above, PPO Online significantly outperforms all of the other methods. To investigate whether PPO Online’s performance is simply due to dataset collected, we fine-tune our BC agent on the PPO Online dataset. We do ablations where the data used for training is from the last 50, 25 and 10 rounds of data collection for the PPO policy. We choose to do this ablation because we expect the quality of the PPO policy performance increases in the later rounds of data collection.

	BC	Complete	Last 50	Last 25	Last 10	PPO Online
reward	0.112	0.201	0.17	0.189	0.235	0.814
percent victories	26.233	38.636	37.023	40.558	41.271	88.4
percent illegal	0.967	1.165	1.159	1.213	1.175	0.722
episode length	12.923	13.21	14.22	14.647	13.338	8.38

Table 8: Comparison between PPO Online and BC agents fine-tuned on the dataset collected by PPO during training. We chose to train on the complete PPO dataset, the last 50 rounds, last 25 rounds, and last 10 rounds of data collected. PPO Online performance still far surpassed performance of the BC agents trained on the PPO policy dataset. Furthermore, there is no substantive difference between training on the complete PPO dataset and the PPO dataset collected in the last 10 rounds.

E.4. Wordle

To evaluate Wordle, we rollout 4096 trajectories against the environment and report the average reward across all the rollouts.

F. Baseline Details

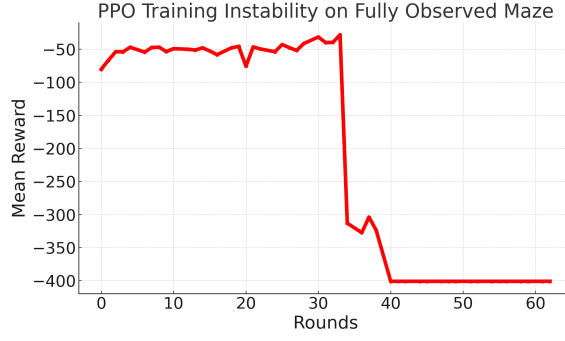


Figure 4: An example of an observed PPO training instability on the fully observed maze task.

F.1. MC Details

The target for these heads is the discounted return-to-go:

$$R_t = \sum_{i=t}^{T-1} \gamma^{i-t} r_i \quad (2)$$

and we use MSE loss for the Q head:

$$J(Q) = \mathbb{E}_{(s_t, a_t, r_{1:T-1}) \sim \mathcal{D}} [(Q(s_t, a_t) - R_t)^2] \quad (3)$$

$$(4)$$

where \mathcal{D} represents the dataset. In MC, $Q(s_t, a_t)$ represents how much more rewards the policy will get if it takes action a_t at the state s_t under some policy (in this case the policy that collected the dataset).

During rollout, when sampling, we perturb the base BC policy with the learned value-functions (Snell et al., 2022a). Let π_β represent the policy trained with BC, and α represent a scalar multiplier, then:

$$\pi_{MC}(a_t | s_t) \propto \pi_\beta(a_t | s_t)^{\alpha Q(s_t, a_t)} \quad (5)$$

F.2. PPO Details

PPO Implementation Details Our PPO implementation uses a learned value function to estimate an advantage baseline. Our value function is fit using GAE (Schulman et al., 2018) value estimates and is implemented as a linear head on top of the same transformer as the policy. We apply clipping to both the importance weights and the value function, as is done in (Castricato et al., 2023). We also apply a KL penalty to the reward, as is standard for RLHF (Stiennon et al., 2022). On some of our tasks, we add a supervised learning, BC, loss term to the standard PPO loss to prevent the model in distribution; this is similar to the approach taken in (Ouyang et al., 2022), which adds a pretraining loss term to the PPO loss to mitigate performance regressions on some benchmarks during RLHF.

PPO Instabilities In some cases we observed training instabilities with PPO in which the policy’s performance would increase for a little bit and then collapse (see Figure 4). We primarily observed this on our maze environment. We are uncertain what the cause of this dramatic instability is, and leave further investigation of this phenomenon to future work.