Extended Abstract

Motivation We are interested in the challenges many traditional RL algorithms face in sparse-reward environments, especially when long-term planning is required. We investigate the effectiveness of offline RL algorithms relative to online algorithms which often fail in these sparse-reward environments. We also aim to explore the differences between different offline algorithms, specifically DPO and SFT.

Method Our environment is Microsoft's TextWorld, which provides a framework for automatically generating randomized text-based games with very sparse rewards. We generated approximately 3000 unique cooking training games and used the provided oracle trajectories as training data. In order to investigate learning strategies in this setting, we employ three different techniques – SFT, DPO, and PPO – and evaluate them separately and in combination with one another. We evaluate the performance of our agents trained under each paradigm on a separate set of 500 test games and report the success rate (number of games completed), average score (overall average non-zero rewards obtained per game), and average steps (average trajectory length).

Implementation To interface with the TextWorld environment for our purposes, we wrote our own module called textworld_utils, which exposes functions to efficiently generate TextWorld games at scale. We obtained expert preference data using the built-in oracle trajectories for each game provided by the TextWorld library. We use that data to finetune the pre-trained LLama 3.2-3B. We ablate with 5 different models: SFT, DPO, SFT+DPO, SFT+DPO+PPO, PPO. We intentionally chose a low parameter count base model for novelty since much larger models have already been shown to do well in TextWorld. We evaluate the performance of each trained agent using our textworld_utils module on a standardized set of 500 systematically generated test games and implemented logging for the success metrics mentioned above.

Results During evaluation, our trained models act as the agent to play all of the games in our testing set, receiving a score for each. Evaluated on our test set of 500 games, we found that the baseline LLaMa-3.2-3B agent results in long trajectories that are unsuccessful. This performance is significantly improved by the application of SFT, as the fine-tuned agent becomes able to solve more than 1/3 of the games in our test set with a far lower average trajectory length. While DPO alone only marginally improves the agent's success rate, it does result in significantly lowered average trajectory length, and training with SFT + DPO retains SFT's performance gains while making the trajectories more efficient (shorter). Our PPO-trained agent obtains a lower-than-baseline average score, but reduces the average trajectory length (although less so than DPO). Finally, our SFT + DPO + PPO trained agent achieves basically the same success rate than SFT + DPO, with a slight improvement on average score and reduced average trajectory length.

Discussion Our results suggest that different training paradigms are better suited to long-horizon, reward-sparse tasks than others. As suggested by theoretical intuition, the offline preference-based algorithms performed much better than PPO in this setting. We also noted through our ablation analysis of SFT and DPO that SFT outperforms DPO in terms of success rate and average score, suggesting that DPO in isolation did not encourage the model to take advantageous actions. However, when the agent was first finetuned with SFT and then finetuned further using DPO, we saw the agent become more efficient. This reinforces the theoretical notion that, while DPO is not very useful to teach a model a task from scratch, it can be useful to refine model responses and decrease the frequency of sub-optimal responses once the model already has a sufficient success rate.

Conclusion We studied the effectiveness of offline preference-based learning in sparse-reward, long-horizon environments using TextWorld as a testbed. Our experiments demonstrate that SFT yields substantial gains in success rate and average reward, outperforming both the baseline and PPO. DPO, while weaker in isolation, improves trajectory efficiency when applied after SFT, validating its utility as a second-stage fine-tuning method. PPO underperforms across all metrics, highlighting the limitations of online RL in sparse-reward settings without many steps. Our results reinforce the theoretical expectation that offline methods paired with high-quality training data are better suited and more efficient than online methods for environments with sparse, hard-to-specify, or delayed rewards. We also happily find the great strength of smaller LLMs in these difficult environments.

Cook or be Cooked: The Bitter Lesson

Derek Askaryar
Department of Computer Science
Stanford University
askaryar@stanford.edu

Parthav Shergill
Department of Computer Science
Stanford University
parthav@stanford.edu

Christy Thompson
Department of Computer Science
Stanford University
cthomps@stanford.edu

Sam Wondsen
Department of Computer Science
Stanford University
swondsen@stanford.edu

Abstract

Sparse-reward, long-horizon environments remain a major obstacle for reinforcement-learning agents, especially when the action and observation spaces are expressed in natural language. We study this setting through Microsoft TextWorld and ask whether offline preference-based techniques can train compact language-model agents that rival far larger systems. Leveraging a scalable generation pipeline, we synthesize 3000 procedurally-generated cooking games for training and 500 disjoint games for evaluation, each accompanied by oracle winning trajectories. From these games, we build a dataset of 6,300 expert trajectories and preference pairs for Supervised Fine-Tuning (SFT) and Direct Preference Optimization (DPO); we also benchmark an online PPO baseline. All methods fine-tune the 3 B-parameter LLaMA 3.2 model. SFT alone raises the zero-shot success rate from 0% to 37% and cuts average trajectory length six-fold. DPO in isolation offers limited gains, but when applied after SFT it further trims trajectories by 11% without sacrificing success. PPO struggles to discover reward from scratch, yet provides a marginal score boost when appended to the SFT + DPO pipeline. Our results show that a lightweight model, trained entirely offline on expert demonstrations and contrastive preferences, can solve over one-third of unseen TextWorld tasks while remaining action-efficient—outperforming much costlier online exploration. The study highlights offline preference learning as an effective, compute-conscious strategy for long-horizon language environments and suggests a promising path toward small-model alignment in sparse-reward domains.

1 Introduction

In this work, we investigate the performance of offline preference-based methods in sparse-reward, long-horizon environments, with a focus on Microsoft's TextWorld, which is a benchmark suite for text-based interactive fiction games. These games present a combinatorially large action space, partial observability, and minimal external reward, making them a compelling evaluation setting for long-term planning and efficient learning.

We compare three paradigms for training language-model-based agents in this setting:

- **Supervised Fine-Tuning (SFT)** using expert demonstrations or high-quality trajectories.
- **Direct Preference Optimization (DPO)** trained on ranked trajectory pairs from expert or model-generated comparisons.

- **Proximal Policy Optimization (PPO)** as a baseline online RL algorithm.

To generate training and evaluation data, we develop a scalable game generation and evaluation pipeline using a custom utility layer over the TextWorld API. We also explore two strategies for collecting preference data: expert rollouts using the environment's built-in oracle, and synthetic comparisons generated using an LLM (Claude via AWS Bedrock). For all approaches, we fine-tune a compact 3B-parameter LLaMA 3.2 model, allowing us to study knowledge distillation and generalization under compute-constrained conditions.

Our experiments show that SFT significantly improves success rates and reward acquisition, while DPO enhances action efficiency when used after SFT. In contrast, PPO struggles to learn effective behavior in this sparse-reward regime. These results highlight the value of offline preference-based methods in long-horizon settings and motivate further exploration of hybrid approaches to sequential decision-making under limited supervision.

2 Related Work

Sparse-reward, long-horizon environments pose fundamental challenges for reinforcement learning algorithms. In such settings, rewards are either delayed or infrequent, making credit assignment difficult and hindering exploration. These limitations are amplified in language-based environments, where the action space is combinatorially large and observations come in the form of natural language. Our work builds on and contributes to four core areas in the literature: early reinforcement learning in text environments, the emergence of controlled benchmarks, preference-based learning methods, and the limitations of online RL in sparse-reward domains.

2.1 Reinforcement Learning in Text-Based Environments

One of the earliest and most influential attempts to apply RL in language environments came from Narasimhan et al., who proposed the Deep Reinforcement Relevance Network (DRRN) (Narasimhan et al. (2015)). In this model, observations and actions are represented as separate embeddings and scored via a relevance function. Although effective on simple interactive fiction tasks, DRRN struggled with exploration in environments where success requires long-term planning. This work illuminated a key difficulty in text-based RL: the agent must interpret a natural language description of the world, decide on a valid action (also in natural language), and maintain memory of past states — all without frequent reward feedback.

These limitations also motivate the need for environments where learning progress can be more rigorously measured and controlled.

2.2 Benchmarks and the Role of TextWorld

To support reproducibility and experimentation in text-based RL, TextWorld was introduced by Côté et al. as a gym-style environment for generating structured interactive fiction games. TextWorld provides the ability to procedurally generate thousands of games with controlled properties (e.g., number of required actions, object complexity), making it an ideal testbed for research in sparse reward learning and long-horizon planning.

One of the earliest successes using TextWorld came from the introduction of KG-A2C, a model that augmented Advantage Actor-Critic with a dynamic knowledge graph and graph attention mechanism. This design enabled the agent to track world state and relational structure more effectively, resulting in more than a 2× improvement on cooking games compared to vanilla A2C. These gains demonstrated the importance of structured representations in solving text-based tasks with delayed rewards and sequential dependencies.

Most recently, advances in large language models have reshaped the landscape. In 2024, a prompt-engineered GPT-4 model with constrained decoding achieved nearly 70% success on the TextWorld CommonSense benchmark — a significant leap in performance. These results suggest that large pretrained models, when guided correctly, can exhibit strong planning and reasoning capabilities in sparse-reward environments, even without explicit RL training.

However, these gains come with trade-offs: models like GPT-4 are computationally expensive to deploy and fine-tune. A key open question — and one central to our work — is whether similar performance can be achieved using smaller models, particularly by leveraging distillation and preference-based offline training from larger model outputs. Our project explores this hypothesis by using a small model (LLaMA 3.2-3B), aiming to replicate some of GPT-4's capabilities more efficiently.

2.3 Preference-Based Learning and Offline Fine-Tuning

When rewards are difficult to define or learn from directly, alternative supervision strategies are needed. One such approach is preference-based learning, where models are trained to prefer one behavior over another. Ziegler et al. demonstrated that large language models could be fine-tuned using preference data to align better with human intent (Ziegler et al. (2020)). Their method collected pairwise human judgments between model outputs and trained a reward model that was then used to fine-tune the base model with reinforcement learning. This became a foundational technique in human feedback alignment and later informed models like InstructGPT and ChatGPT.

More recently, Rafailov et al. proposed Direct Preference Optimization (DPO) (Rafailov et al. (2024)), which bypasses reward modeling entirely. Instead of fitting a reward model and training a policy on it, DPO directly updates the model to assign higher likelihood to preferred responses. This makes it well-suited for offline, data-efficient training in settings like ours.

In our project, we explore the interplay between SFT and DPO. While SFT gives the model a general understanding of task structure and valid action patterns, DPO serves to refine its responses, reducing suboptimal behavior and improving trajectory efficiency.

2.4 Online RL in Sparse-Reward Environments

As a point of comparison, we evaluate our models against a PPO-trained model, a widely used online RL algorithm known for its robustness and ease of implementation (Schulman et al. (2017)). PPO has shown success in many domains, but it assumes frequent, well-shaped rewards to guide learning. In sparse environments like TextWorld, we find that PPO often fails to learn meaningful behavior from scratch, and forebodes needing orders of magnitude more steps to show some learning. Even when it slightly reduces trajectory length, it underperforms on success rate and total reward as a baseline, confirming that online exploration-based learning by itself is not competitive in environments with minimal intrinsic reward signals.

These findings echo broader themes in the literature: that offline learning from curated data — particularly preference-based signals — can offer more effective supervision than reward maximization alone in complex, reward-sparse environments.

3 Method

Our goal is to evaluate the effectiveness of preference-based offline learning strategies in sparse-reward, long-horizon environments. We use Microsoft's TextWorld environment to benchmark three learning paradigms: Supervised Fine-Tuning (SFT), Direct Preference Optimization (DPO), and Proximal Policy Optimization (PPO). This section details our game generation pipeline, data preparation strategies, model architecture and prompting, training procedures, and evaluation protocol.

3.1 Environment: TextWorld Cooking Games

We use the TextWorld gym environment to generate thousands of procedurally generated cooking-themed games. Each game defines an end objective, such as preparing and eating a specific dish, and unfolds as a sequence of natural language observations and action commands. To win, you must execute the right commands in order such as "fry onion", "prepare meal", and "eat meal". The environment is only partially observable at the beginning and must be explored. Players only receive sparse reward signals. Only the commands in the winning trajectory yield positive reward. The number of required commands to win depends on the difficulty settings used to generate that specific game, but usually not many commands are needed. However, it may take many exploratory commands aside from the winning commands to reach the winning state and those exploratory commands give no reward feedback. Trajectories can get long with rewards few and far in between making this a canonically difficult RL environment.

3.2 Game Generation and Data Collection

Using Microsoft's TextWorld Python library, we implemented our own module called `textworld_utils` to automate the generation and evaluation of game instances. This tool supports:

- Procedural generation of thousands of training and test games with fixed or randomized seed control, and various generation flags to change game difficulty: These flags represent things that are required to finish the recipe for that game. For example, generation flags include cutting, cooking, number of ingredients that must be collected for the recipe, and more. The more flags that are enabled usually means more commands needed to win. Additionally, the adjective-noun pairs of food ("fried fish", "chopped chicken", etc) are split into three different distributions which are train, valid, and test. The train and test games we generated were generated using their respective distributions which allows us to better test the generalization ability of our learned agents.
- Environment rollouts using oracle and learned agents.
- Logging of observations, admissible commands, actions taken, rewards received, and ground-truth objectives.

We use this tool to construct:

- A training set of approx. 3,000 games each generated with different random seeds and different combinations of difficulty flags so the agent can learn many different difficulty levels of games.
- A test set of 500 unseen games for final evaluation.

3.3 Data Collection for Offline Algorithms

We collect two forms of training data:

Expert Demonstrations for SFT Collecting expert demonstrations for the SFT fine-tuning was quite simple because TextWorld games that generated from the training distributions actually come with the "oracle", or fastest winning, trajectory of commands. We were then able to create a dataset of expert trajectories recorded as (observation, admissible commands, oracle command) tuples.

Action Preferences for DPO To support DPO, we construct preference pair data point in the form of (observation, chosen action, rejected action) tuples. These are generated by using the TextWorld oracle optimal action as the "chosen" action, and a random incorrect action as the "rejected" action.

An interesting note to add is that since previous work in the TextWorld environment was fairly successful by using very large models such as GPT-4 or Claude we wanted this paper to train a much smaller model and conduct knowledge distillation by using Claude to generate synthetic trajectory data. However, we quickly found that Claude was actually not very good at TextWorld and the number of tokens needed for the input prompts were very costly. Therefore, we used the TextWorld oracle trajectories as a proxy for an expert who we are conducting knowledge distillation on. Overall, we had a total of 6,300 expert trajectories. Each game-state in the trajectory is accompanied by the corresponding oracle-generated optimal action (used for both SFT and DPO), and a randomly-chosen non-optimal action (used only for DPO).

3.4 Prompting and Model Interface

For all prompting and training involving LLMs (SFT and DPO), we use a consistent structured prompt template designed to:

- Summarize the overall game goal.
- Include recent history as observation–action pairs.

- Display the current observation and admissible commands.

In terms of the actual request for completion, the agent is asked to:

> "Choose a NEW action from the 'Available commands' list that will help you complete your objectives... Your response MUST be **exactly one** command... DO NOT explain your choice."

This format enforces consistent input formatting and response constraints for efficient token-level supervision. The same prompt is used for fine-tuning and preference evaluation to ensure consistency.

3.5 Evaluation Protocol

We evaluate all agents on a standardized set of 500 held-out test games. For each game, we compute:

- **Success Rate:** Percentage of games in which the agent achieves the full 4.0 score.
- **Average Score:** Mean total reward over all test games.
- **Average Trajectory Length:** Mean number of steps per episode, truncated at environment max length (more than sufficient to win games).

Evaluations are run using our `textworld_utils` module, with deterministic seeds and consistent command filters. Each agent's outputs are logged and parsed into trajectory metrics. We further conduct ablation studies by selectively disabling SFT or DPO during training to isolate their contributions. In total, we have 4 experimental models, excluding the baseline.

4 Experimental Setup

All agents are fine-tuned on the **LLaMA 3.2–3B** model, chosen for its lightweight footprint and suitability for distillation and preference tuning. All models share a decoder-only, autoregressive architecture with frozen tokenizer and vocab.

SFT Using Hugging Face's TRL library, we fine-tune the base model on expert demonstration data using cross-entropy loss over the target action token in the prompt-completion format. We train for multiple epochs with early stopping on held-out trajectory data.

DPO We use Hugging Face's implementation of Direct Preference Optimization. Each training step samples a preference pair and applies the DPO objective to maximize the margin between the log-likelihood of the preferred and rejected completions. We use DPO in two experiments. For the first, we just fine-tune base LLaMa 3.2 using DPO and our preference dataset. For second, we apply DPO on top of an already SFT-fine-tuned LLaMa. We do this to see if two offline algorithms can be combined to further improve model performance.

PPO Finally, We use a custom PPO training system for Llama 3.2-3B-Instruct on TextWorld cooking games. Our approach uses an LLMValueWrapper that extends the base language model with a learned value head for critic functionality. The policy leverages the language model's generation capabilities through batched token probability ranking, where action selection is performed by computing log-probabilities for each admissible command and sampling from the resulting distribution. Training involves collecting rollouts through environment interaction, computing advantages using Generalized Advantage Estimation (GAE), and performing multiple epochs of PPO optimization with clipped policy gradients. Unlike approaches that require auxiliary shaping or demonstrations, our method trains directly from environmental rewards using the native TextWorld API.

5 Results

5.1 Quantitative Evaluation

We evaluated all trained agents on a held-out set of 500 games from the TextWorld test distribution. These games were sampled from a distribution disjoint from the training set, both in terms of random seed and recipe components. Table 1 summarizes the key metrics: success rate (percentage of games completed with a perfect score of 4.0), average score (total reward per game), and average steps (trajectory length until termination or max limit). We also conducted an ablation to examine the impact of dataset size on SFT performance (Table 3).

Table 1: Evaluation results on 500 held-out TextWorld test games.

Model	Success Rate	Average Score	Average Steps
Baseline (LLaMA-3.2-3B)	0.00	1.13	56.30
SFT	**0.37**	1.88	9.96
DPO	0.06	1.43	11.79
PPO	0.00	0.43	17.30
SFT + DPO	0.36	1.91	8.88
SFT + DPO + PPO	0.36	**1.94**	**8.84**

Key Findings.

- **Baseline model** performed poorly, with zero success rate and long average trajectories. This shows the difficulty of the environment and the need for supervision or reward shaping.
- **SFT** significantly improved performance, solving over a third of all test games and drastically reducing trajectory length. This supports the hypothesis that direct supervision using oracle actions enables meaningful learning in sparse reward environments.
- **DPO** alone yielded marginal improvement in success rate and average score over the baseline, but reduced average trajectory length. This suggests DPO encourages more efficient actions even if success is not frequently achieved.
- **PPO** in isolation performed worst across all metrics except average steps, confirming that online RL with sparse rewards is ineffective without extensive reward shaping or auxiliary objectives.
- **SFT + DPO** performed best overall in terms of trajectory length, with close to the highest success rate and average score. This supports the idea that DPO is more effective when used as a second-stage refinement after SFT.
- **SFT + DPO + PPO** performed similarly to SFT and SFT + DPO in terms of success rate and and average steps, with a slight improvement in average score. This supports the notion that PPO is not very effective on its own in a sparse-reward setting like TextWorld, and the bulk of performance gains come from the application of SFT and DPO, but with a much greater scaling of steps PPO could show marked improvement, as shown by the slight improvements in average score and steps over the SFT + DPO model here.

5.2 Qualitative Analysis

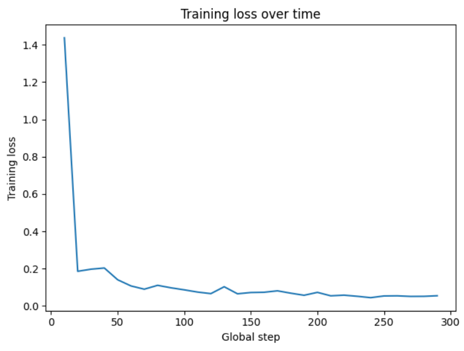
Game ID	Steps	Score	Key Behavior and Notes
1 (Success)	4	3.0	Efficient minimal plan: opened fridge, took cilantro, prepared meal, ate meal. No unnecessary actions.
2 (Success)	4	3.0	Similar to Game 1. Took correct ingredient (red onion), prepared and ate meal in optimal steps.
0 (Failure)	3	1.0	Took yellow bell pepper, cooked it without checking recipe. Did not prepare or eat, terminated early.
499 (Failure)	80	0.0	Long, degenerate episode. Repeatedly dropped/picked objects, cooked random items (burned bell pepper, ate onion), examined fridge/cheese 30 times. Never prepared a valid meal.

Table 2: Qualitative Examples of SFT + DPO + PPO Agent Behavior in TextWorld

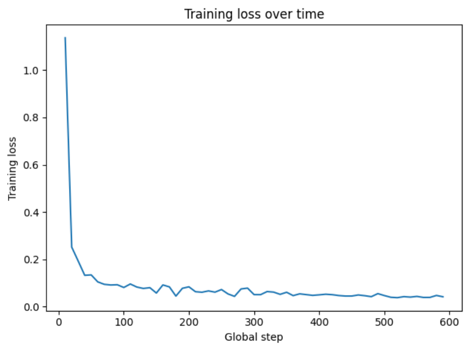
These qualitative examples serve to demonstrate the ceiling our model is currently hitting. It can successfully execute short, linear recipes when it comes to opening the fridge, picking an ingredient, preparing it, and then eating it. However, it also tends to either fail sooner than the length of a win or fail much later. This is due to behavior exemplified by the two failure cases, the first of which demonstrates a breakdown in the correct order of actions leading to early termination. It still gets a point for taking the yellow bell pepper from the fridge, but it cooks it on the stove without preparing it. The second example demonstrates the other frequent fail case with our current agent, where it goes on forever due to earlier actions. On multiple occasions, the agent falls into loops of actions, onset by first frying the block of cheese, then taking it and dropping it because of the observation that the block is now fried being different from what it expects. The repeated examination of the cheese and the fridge spanning dozens of steps suggests the agent becomes fixated on an invalid plan with no internal mechanism to redirect its behavior. In both cases, failure is not due to a lack of individual action competence, but rather a failure in sequencing and goal inference. Although great bounds were made to generalize this far, gaining greater success would most likely reside in using richer preferences in a multi-stage alignment process.

6 Discussion

6.1 SFT Discussion



(a) SFT Training Loss on Half Dataset



(b) SFT Training Loss on Full Dataset

Figure 1: SFT Training Loss

As we see from the results above, SFT by itself seems to provide the most performance gains in all metrics compared to both DPO and PPO in isolation. However, there are interesting things to note. For one, we see from Figure 1 that the training loss curve for SFT converges to its minimum very quickly, within almost only 50 steps of our 600 step training episode. This could mean many things, but one thing it tells us is that the base LLaMa 3.2 model might already have most of the intelligence needed to solve these TextWorld games, but it just needed to be fine-tuned very slightly on the task with just some prompt-completion expert pairs. We also decided to run another experiment whether we trained LLaMa on half the data (only 3,150 expert trajectories) to see how much the diversity and size of the dataset mattered. The results are shown in Table 3.

Table 3: Effect of training dataset size on SFT agent performance.

Training Set Size	**Success Rate**	**Average Score**	**Average Steps**
Full (6,300 trajectories)	0.36	**1.916**	**9.96**
Half (3,150 trajectories)	**0.368**	1.876	12.60

The results reveal that halving the dataset size has minimal impact on success rate and average score, though it increases trajectory length. This suggests that SFT is relatively robust to moderate reductions in data volume and that smaller, high-quality datasets may be sufficient in constrained settings. Additionally, it may signify the high level of intelligence of the base LLaMa model.

6.2 DPO

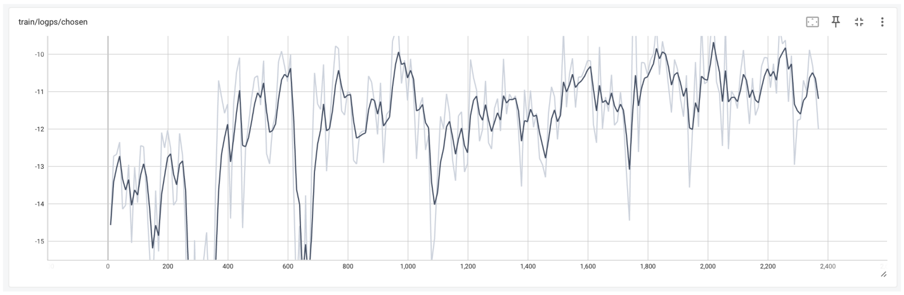
The poor performance of isolated DPO was initially surprising to us, but makes sense given its actual loss function given here:

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\log\sigma\left(\beta\log\frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \beta\log\frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)}\right)\right]$$

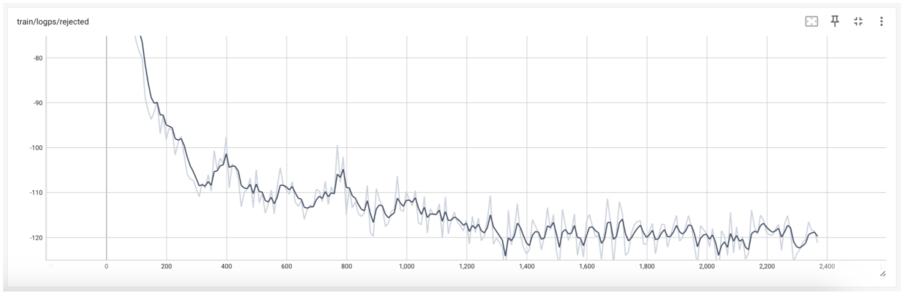
where:

- π_θ is the language model we want to fine-tune.
- π_{ref} is the reference (frozen) model.
- $\mathcal{D}$ is the dataset of preference tuples (x, y_w, y_l).
- x is the prompt.
- y_w is the preferred response.
- y_l is the dispreferred response.
- β is a temperature hyperparameter that controls the regularization strength.
- σ denotes the sigmoid function.

From the function, we see that DPO does not necessarily increase the log-prob of the preferred response, or chosen action, but rather just wants to increase the distance in log-probs of the chosen and rejected actions. We graph the log-probs of the chosen and rejected actions over the course of training in Figure 2. From these curves, we see that our training achieves the DPO objective by decreasing the logprobs of the rejected actions and only slightly increasing, if at all, the log-prob of the chosen actions. This explains the model's poor performance because it isn't really learning what is most optimal, rather it is just learning what is not optimal which is a much more indirect way of learning. Since the log-probs of the chosen actions were likely not high enough to begin with, the agent never really gets better. This also explains the very choppy training loss curve for isolated DPO.



(a) Log Probability of Chosen Actions



(b) Log Probability of Chosen Actions

Figure 2: Chosen vs. Rejected Log Probabilities

6.3 SFT + DPO

Our most successful model was first fine-tuned with SFT and then with DPO. SFT obviously provided the vast majority of the performance gains, but DPO was still able to further refine the model with the most significant benefit being a decrease in the average number of steps to finish a game. We theorize that DPO was able to do so by decreasing the log-probs of the rejected actions, as seen in Figure 2, which made it so the model was less likely to take suboptimal actions and therefore the model more often took the quickest winning trajectory rather than a longer winning trajectory.

6.4 PPO

Finally, the PPO-only trained model does the poorest and does not improve on the baseline besides average steps. This is, however, completely expected because PPO is an online algorithm which heavily depends on the reward signals received throughout the games which are extremely sparse in TextWorld. With this level of sparsity, the agent has very little hope to learn and improve without an enormous amount of steps.

6.5 SFT + DPO + PPO

As a way of validating our intuition about the relative efficacy of SFT + DPO vs. PPO, we evaluated another agent fine-tuned with PPO after SFT + DPO. We found that this agent performs very slightly better to our SFT + DPO agent, which is expected given the fact that PPO's online learning does not yield a significant additional learning benefit in sparse-reward long horizon settings like TextWorld. However, as mentioned before it would be interesting to see a greater resource project take on using PPO for orders of magnitude more steps to see if it could actually capitalize off of the small improvements we saw.

6.6 Difficulties

Working with the TextWorld was uniquely difficult in many ways. Along the course of the project we had to pivot many times. For one, we originally wanted to do knowledge distillation from Anthropic's Claude model by having Claude play games and saving the trajectories as our expert training data. However, the API requests were prohibitively expensive and Claude was surprisingly bad at playing which made the training data low-quality. We tried to finetune models using the Claude-generated data, but they actually decreased in performance because of the poor training signal. Luckily, we were able to pivot since we found that when TextWorld games are generated the metadata also includes the fastest winning trajectory which is, by definition, great expert data to train with.

Another difficulty that arose during this project was the environment itself. Since TextWorld is text-based, the models must play the game by taking a prompt. Designing and using this prompt was actually extremely nitpicky and difficult. For the model to properly play the game, we had to include the entire game history up to that step in the prompt. Also, we found that we had to include the set of admissible commands in the prompt otherwise the model would frequently give non-commands. All of this information in the prompt then led to the issue of hitting the max content length for the base models. We actually originally trained our models using Gemma, but were getting very low performance. We spent an inordinate amount of time bugfixing until we realized that many of the prompts were getting truncated by the TRL library during training because of Gemma's context length limits. This led to us using LLaMa 3.2 instead which has a context length window of 128K tokens which was many times larger and sufficient for our purposes. Even then, we spent even more time trying to refine the prompt further to try and improve the performance of the models. Our initial iterations on the prompt actually made a significant difference which marks the potential necessity of sufficient prompt-engineering in LLM tasks.

7 Conclusion

In this work, we evaluated the effectiveness of preference-based offline learning in sparse-reward, long-horizon environments using the TextWorld benchmark. Our experiments demonstrate that offline methods, specifically those leveraging expert trajectories, can significantly outperform online reinforcement learning approaches like PPO. Supervised Fine-Tuning (SFT) on oracle data yielded the most substantial improvements in success rate and reward acquisition. Direct Preference Optimization (DPO), when applied after SFT, served as an effective refinement step, improving trajectory efficiency by reducing suboptimal actions.

We found that DPO alone was less effective, likely due to its contrastive objective that prioritizes the suppression of bad actions rather than direct optimization of good ones. PPO, when used in isolation, underperformed across all metrics, reinforcing the difficulty of online exploration in sparse-reward domains. However, when applied after SFT and DPO, PPO yielded modest additional gains in average score and retained the improvement in success rate and trajectory length. This suggests a limited but complementary role in further fine-tuning a warm-started policy.

Our findings underscore that combining structured supervision (SFT) with preference-based refinement (DPO) forms a highly effective offline training pipeline. Even with a lightweight 3B-parameter model like LLaMA 3.2, we achieved strong generalization and behavioral efficiency with minimal online interaction. These results highlight the promise of small-model alignment via distilled expert knowledge and offline preferences, offering a practical and scalable path forward for RL in sparse, long-horizon domains.

Future work may explore multi-stage alignment using richer preferences, dynamic curriculum learning, or extensions to other high-dimensional environments such as embodied agents and interactive dialogue.

8 Team Contributions

- **Derek Askaryar:** DPO Implementation, Game Generation
- **Parthav Shergill:** TextWorld Utils Implementation, Evaluation Pipeline
- **Christy Thompson:** SFT Implementation, Preference Data Collection
- **Sam Wondsen:** PPO Implementation, Qualitative Response Analysis

Changes from Proposal We used the TextWorld environment, as opposed to 2048, because rewards are much sparser and game-playing is based on natural language output. We also used the SFT, DPO, and PPO algorithms instead of self-alignment, since we were interested in ways that models learn *without* a reward framework.

References

Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. 2015. Language Understanding for Text-based Games Using Deep Reinforcement Learning. arXiv:1506.08941 [cs.CL] `https://arxiv.org/abs/1506.08941`

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2024. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. arXiv:2305.18290 [cs.LG] `https://arxiv.org/abs/2305.18290`

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. arXiv:1707.06347 [cs.LG] `https://arxiv.org/abs/1707.06347`

Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2020. Fine-Tuning Language Models from Human Preferences. arXiv:1909.08593 [cs.CL] `https://arxiv.org/abs/1909.08593`