Language Model-In-The-Loop: Data Optimal Approach to Recommend Actions in Text Games

Anonymous authors
Paper under double-blind review

Abstract

Large Language Models (LLMs) have demonstrated superior performance in language understanding benchmarks. A recent use case for LLMs involves training decision-making agents over textual information. The existing approach leverages LLM's linguistic priors for action candidate recommendations in text games, i.e., to operate without environment-provided actions. However, adapting LLMs to specific games/tasks requires a massive amount of annotated human gameplay. Moreover, in the existing approach, the language model was kept frozen during an agent's training process, which limits learning from in-game knowledge about the world. Hence, we explore strategies to adapt the language model for candidate recommendation with in-game transition in an online learning fashion to mitigate reliance on human-annotated gameplays, which are costly to acquire. In this paper, we propose in-game transition selection methods to adapt the LLM in the loop, reducing the dependency on using human-annotated gameplays while improving performance and convergence. Our method demonstrates a 53% relative improvement in average game score over the previous state-of-the-art model, achieving more than twice the convergence rate in a full-annotated dataset setting. Furthermore, even with only 10% of human annotation, we surpassed the 100% state-of-the-art performance benchmark.

1 Introduction

Large Language Models (LLMs) (Devlin et al., 2019a; Radford et al., 2018b; Ouyang et al., 2022) trained on large corpora of unstructured text corpora are the state-of-the-art models in several Natural Language Understanding (NLU) benchmarks. Bender & Koller (2020) argue in their position paper that the models mainly trained from static benchmarks rely on the *form* rather than understanding the meaning. Also, there has been a recent interest in interactive training of large language models in situated learning environments. Bisk et al. (2020); McClelland et al. (2020) point out the necessity for LMs to have enhanced language understanding and meaning through interacting with the physical world. Also, Lake & Murphy (2021) argues that LMs fall short in their communicative usage, requiring reasoning over intents despite their success in static datasets.

Training decision-making agents over textual information for playing text-based games (Hausknecht et al., 2020; Côté et al., 2018) has been a recent use case for LLM. While decision-making has been the front of text-game playing, such games introduce novel challenges for language understanding and domain adaptation for LLMs. Jerchio (Hausknecht et al., 2020) is a collection of text-based games; a sample gameplay from "Zork1" is illustrated as in Figure 1. An agent receives a textual observation about its environment that it has to understand and reason over the possible actions to pick one and proceed. The Zork1 game has a vocabulary size of 697 and has approximately $697^4 \sim 200$ billion potential 4-word actions. Such a setup allows for qualitatively understanding the LLM's abilities to *understand*, *reason*, and *adapt* to novel situations.

To handle combinatorially large action space, CALM (Yao et al., 2020) introduced a dataset with a corpus of human gameplay on similar games called `ClubFloyd` to finetune the GPT-2 to generate candidate actions. Then, these actions are used by the decision-making agent called the Deep Reinforcement Relevance Network (DRRN) (He et al., 2016) on the Jericho benchmark (Hausknecht et al., 2020)–a suite of text-based games–. After the initial adaptation to the human-annotated corpus, the language model remains frozen throughout the learning within the game. Further, they observed that the performance of the text-based games in the Jericho benchmark is proportional to the size of the annotated human gameplay corpus; such reliance adds to the cost and makes it hard to transfer this approach to other problem settings.

Figure 1: Sample gameplay from the Zork1 game in Jericho using LM for action recommendation: LM recommends action candidates based on observations from the environment. The RL agent selects an action from the candidates.

On the one hand, there is a need to mitigate the reliance on human-annotated transitions to scale applications of LLMs. On the other hand, in-game transitions remain unutilized for training the LLM. Although one can use the transitions to train the model, the solution requires a comprehensive analysis of what such an LM-in-the-loop training entails.

Toward that goal, we study different strategies to adapt an LM in an online fashion. We use a buffer to store in-game transitions during training to collect several data points along the timesteps. The reason for this buffer is to enable batched updates and reduce the stochasticity of the LM updates. We employ diverse sampling techniques to sample data from the buffer to adapt the language model. Further, we analyze such a setup along three main dimensions: (1) Performance, (2) Convergence rate, and (3) Reliance on human-annotated transitions.

The main contributions of this work are summarized as follows:

- Proposed a framework for adapting language models for action suggestions through in-game-generated transitions.
- Explored different approaches to adapting the language model with in-game transitions.
- Adapting LM using state feature-based transitions selection provided more significant performance gains and faster convergence over the state-of-the-art performance benchmark.
- LM-in-the-Loop reduces the emphasis on human-annotated transitions and enables accelerated convergence.

2 Related Work

Text Games: Jericho (Hausknecht et al., 2020) is a popular learning environment that supports 32 human-written interactive fiction games. These games are designed to be difficult for human players, serving as a more realistic training ground to evaluate language understanding agents. Compared with frameworks like TextWorld (Côté et al., 2018), these games have significantly more linguistic variety and larger action space. Jericho environment provides a smaller list of candidate actions that can be used to train reinforcement learning (RL) agents. Approaches like DRRN (He et al., 2016), TDQN (Hausknecht et al., 2020), and KGA2C (Ammanabrolu & Hausknecht, 2020) have used *handicap* to operate on small action space and learn only through in-game rewards. Towards using large LMs, environment-provided actions are replaced with LM-generated actions like with GPT-2 (Yao et al., 2020), or BERT (Singh et al., 2021).

Transformers in RL: Transformer architectures are now being increasingly used in reinforcement learning (RL); Chen et al. (2021); Janner et al. (2021) use smaller transformer architectures on Atari games that earlier used convolutional networks as policy networks in offline settings. Further adaptations to make the architectures lightweight to enable online training was proposed in Xu et al. (2020); Parisotto et al. (2019); Ouyang et al. (2022); Reid et al. (2022); Tarasov et al. (2022); Ahn et al. (2022). Yao et al. (2020) explore using the semantic prior in GPT-2 for candidate action recommendation in text games. Further, Tuyls et al. (2022); Li et al. (2022) train LMs to remember optimal trajectories to move to novel game regions swiftly.

Data Efficiency: LLMs (Devlin et al., 2019b; Brown et al., 2020) are pre-trained with a tremendous amount of unstructured text data from *the web* using a generic language modeling objective. Adapting the models to downstream tasks(Khashabi et al., 2020; Rajpurkar et al., 2016; Zhang et al., 2015; Maas et al., 2011), however, is greatly affected by the quality of supervision and the dataset size. As reliance on annotated data makes their application hard to scale, techniques like data augmentation Feng et al. (2021), using distilled models Radford et al. (2018a), and learning from toyish data Wu et al. (2022) have been explored as alternatives. In contrast, our approach aims to reduce the need for human annotations to adapt LM in an interactive learning setup.

3 Background

3.1 Text Games

In text-based games, at each step, t, a learning agent interacts with the game environment by generating a textual action $a_t \in \mathcal{A}_t$ that is relevant to the textual observation o_t. The agent receives a scalar reward $r_t = \mathcal{R}_t(o_t, a_t)$. The agent maximizes the expected cumulative rewards $(r_0, r_1, r_2, \ldots r_N)$, until the end of an N-step-long episode.

3.2 DRRN and Advantage Function

A popular deep RL method used in text-based games is the Deep Reinforcement Relevance Network (DRRN) (He et al., 2016). The observation (o) and actions (a) are first encoded using separate recurrent neural network encoders (such as a GRU (Chung et al., 2014)) f_o and f_a, respectively. A decoder g then combines the representations to obtain the Q-value using a network parameterized by Φ:

$$Q^{\Phi}(o, a) = g(f_o(o), f_a(a)). \tag{1}$$

The DRRN learns to estimate the Q-value through iteratively updating Φ with experience sampled from a prioritized experience replay buffer with the temporal difference (TD) loss:

$$\mathcal{L}_{TD}(\Phi) = \left(r + \gamma \max_{a' \in A} Q^{\Phi}(o', a') - Q^{\Phi}(o, a)\right)^2, \tag{2}$$

where r and o' are the reward and the observation received after taking action a upon observing o, and γ represents the discount factor.

Advantage function: An estimate how good an action, a, is when chosen in a state, o, is obtained by subtracting the value of the state ($V(o)$)—a weighted average of the Q-values— from the $Q(o, a)$ of that particular action in that state.

$$A(o, a) = Q^{\Phi}(o, a) - V^{\psi}(o) \tag{3}$$

Q-Value estimates the expected reward after a specific action was played, whereas $\mathrm{V}^{\psi}(o)$ is the parameterized estimate of the expected reward from being in o before an action was selected.

3.3 LLM for Action Recommendation

Consider a dataset $\mathcal{D}$ of N transitions of human gameplay across different games organized in context-action pairs as $((o_{t-1}, a_{t-1}, o_t), a_t)$. For example, a sample could be like, "`[CLS]... to the north is a restaurant where the mayor ate often. to the east is the mayor's home. [SEP] northeast[SEP] ... you are carrying nothing. you are still on the streets. ... [SEP] northeast`". `[SEP]` and `[CLS]` are special tokens specific to LM-training. Yao et al. (2020) uses the ClubFloyd dataset to adapt a pre-trained GPT-2 model with a causal language modeling task. The motivation is to enable the linguistic prior of GPT-2 to adapt to the games and provide better action recommendations to the DRRN.

4 Methodology

4.1 LM-in-the-Loop to recommend Actions

The game-playing agent takes sequence of actions according to the game's rules in the Jericho environment. The environment has two scenarios—with and without *handicap*—which correspond to whether the actions can be generated from within the possible actions suggested by the environment or without any limitations by the environment respectively. The *with handicap* setup evaluates the agent exclusively on planning with the actions provided. In contrast, the *without handicap* requires the agent, in addition to understanding the observation, to generate acceptable candidates.

In CALM (Yao et al., 2020), the LLM is kept constant throughout the gameplay. We use a similar setup for action recommendation as in CALM, where a trained GPT-2 LM is adapted with the Clubfloyd dataset to recommend actions to the DRRN agent. We explore the feasibility, prospects, and challenges that entail training LM-in-the-loop post-finetuning with human gameplays in ClubFloyd adaptation as in Table 1. Towards that, in addition to training the DRRN agent with TD-learning (Equation 2), we collect the transitions $(o_t, a_t, o_{t+1}, r_{t+1})$ throughout the game episode, e^{TD}, and populate them in $\mathcal{D}^+$ and $\mathcal{D}^-$ based on a heuristic that depends on—reward, return, and the game states.

First, with LM parameterized by θ and generating action candidates, we train DRRN for n^{RL} consecutive episodes. After n^{RL} episodes, we sample d^{LM} sized dataset from $\mathcal{D}^+$, and $\mathcal{D}^-$ with probabilities p^+ and $1 - p^+$ respectively for 2000 gradient steps at finetuned after every k game steps. To train LM, we use a weighted cross-entropy loss:

$$\mathcal{L}^{LM}(\theta) = -\mathbb{E}_{(a_t, o_t) \sim (\mathcal{D}^+, \mathcal{D}^-)} \log P_\theta(a_t \mid o_t) \cdot h(\cdot) \tag{4}$$

Then, we plug in back the in-game trained LM to recommend actions for the DRRN agent. The maximum buffer size of $\mathcal{D}^+$, $\mathcal{D}^-$, p^+, d^{LM}, and n^{RL} are all game-specific hyperparameters. The $h(\cdot)$ is defined as a function of reward r_t, or action-advantage, $A(o_t, a_t)$, or assumed 1 uniformly $\forall (o, a) \in \mathcal{O} \times \mathcal{A}$. We evaluate different approaches based on the sampling of transitions, and the loss function ($\mathcal{L}$), used for training the language model. Approaches for LM-in-the-Loop based on the construction of $\mathcal{D}$, and sampling are:

***Un*categorized Transitions (UT):** In this setting the transitions stored in the buffer are not categorized by any special heuristic function. We simplify this approach by maintaining a single buffer, $\mathcal{D}$ and samples are drawn randomly from the buffer $\mathcal{D}$. This is a weaker baseline than other heuristics for selecting useful transitions based on their importance.

***Un*categorized Transitions - Linear weighted Advantage (UT^{LA}) :** In this, the transition data is kept in a single buffer $\mathcal{D}$ similar to in the UT setting. To finetune the language model using the weighted cross-entropy loss (Equation 4), we use the weighted advantage function (Equation 3).

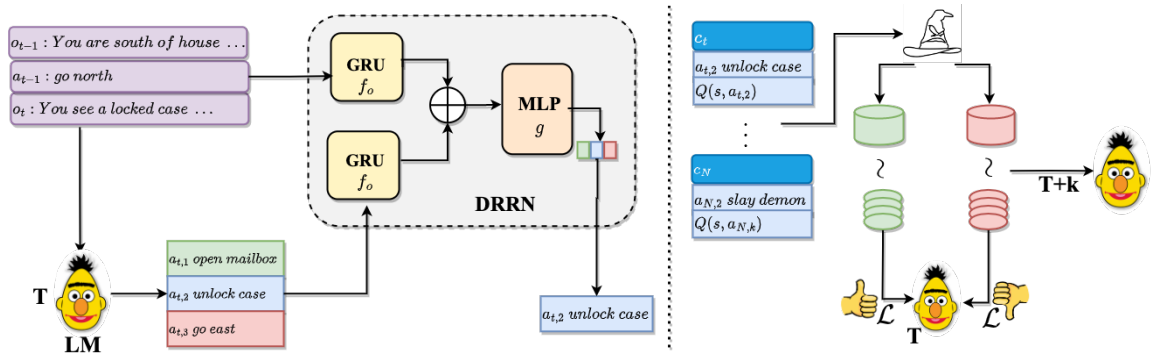


Figure 2: Training LM-in-the-Loop post-human-annotated dataset adaptation: RL agent (DRRN) picks the action the language model recommends (at T), GPT-2. The context pairs are stored in the replay buffers and categorized according to some heuristics. Then, the Language model is updated with in-game transitions after k learning steps in the game. Finally, the updated language model $(T + k)$ actions are recommended.

This variant, UT^{LA}, allows for negative weights $[-\infty, +\infty]$ with $h(\cdot)$ as follows:

$$h(o_t, a_t) = 1 + \beta \cdot A(o_t, a_t), \tag{5}$$

where, $\beta \in \mathbb{R}^+$ is a hyperparameter.

***Un*categorized Transitions - Exponential weighted Advantage (UT^{EA})** : The procedure is very similar to UT^{LA}. However, in UT^{EA} we use exponential weighted advantage function which is strictly non-negative $[1, +\infty]$ using $h(\cdot)$ function:

$$h(o_t, a_t) = e^{\beta \cdot A(o_t, a_t)}, \tag{6}$$

where, $\beta \in \mathbb{R}^+$ is a hyperparameter.

Reward Trajectories (RT): The reward from transitions, r_t, is used to categorize positive and negative trajectories. When $r_t > 0$ all transitions up until the earlier non-zero reward are considered positive and added to $\mathcal{D}^+$. Further, we explore utilizing the return, reward, and advantage function of actions to re-weight $\mathcal{L}^{LM}$ using the $h(\cdot)$ function over **UT** setting as above. We describe them as follows:

State Features (SF): In this, the transitions are labeled as useful or not based on whether an action a_t resulted in reward increase or if the agent's location changed. *i.e.,* moved from one room to another. The location information received is an artifact of the game framework. Further, we vary p^+ to maximize the transitions that encourage exploration to eventually result in improved performance in the game. Here, $h(\cdot)$ is fixed as 1 uniformly $\forall (o, a) \in \mathcal{O} \times \mathcal{A}$.

5 Experiments

We perform comprehensive experiments with LM-in-the-loop setup to study the following questions:

1. Does including the language model in the training loop improve performance?
2. Should the transitions be categorized for improved learning?
3. Does LM-in-the-Loop mitigate the reliance on human gameplay transitions?

5.1 Task Adaptation Dataset

ClubFloyd dataset (Yao et al., 2020) is a collection of crawled data from the ClubFloyd website. The dataset comprises gameplay from experienced players; however, they may not be familiar with the particular games, so the actions are not optimal. This dataset includes 426 transcripts covering 590 unique games; and contains $223,527$ pairs of context and in the form of $((o_{t-1}, a_{t-1}, o_t), a_t)$.

5.2 Benchmark and the Metric

Jericho (Hausknecht et al., 2020) is a learning environment that supports human-written interactive fiction games as described in Figure 1. We chose 10 games based on the diversity in the challenges faced in each game, such as large action space, solution length, and reward sparsity as mentioned in Hausknecht et al. (2020). We use the average of the last 100-episodes' score with standard error for individual games as our metric for evaluation.

In addition, we report the average normalized (avg. norm) score against the maximum score possible in each game, which estimates the human-machine gap in text-based games. Finally, we also report the relative performance percentage difference between the baseline and the best approach mentioned as $\Delta\%$ in Table 1 to capture the improvement as the scores' range in each game differs.

5.3 Model Details

Language model (GPT-2) is first finetuned on the ClubFloyd dataset. Given the context, (o_{t-1}, a_{t-1}, o_t), the finetuned GPT-2 proposes action candidates for DRRN to choose. Following that, every action candidate and context is encoded with a GRU. Then, a decoder combines the representations to estimate the Q-value using a multilayer Perceptron (MLP) and updates the DRRN agent parameter Φ. During the training process of the DRRN agents, the context-action pairs are stored in the replay buffers. After k steps, we sample d^{LM} sized dataset from $\mathcal{D}^+$, and $\mathcal{D}^-$ with probabilities p^+ and $1-p^+$ respectively and update the language model with in-game transitions. Then, the updated language model is used to propose the action candidates.

The buffer size is $100K$ for replay buffers that use the First-In-First-Out (FIFO) strategy to replace samples. To train, d^{LM} samples are sampled uniformly randomly from the two buffers D^+ and D^-. However, the probability of choosing the buffers is defined by p^+ and p^- $(1-p^+)$, respectively. The number of gradient steps for LM training is fixed at 2000 across the setups. And, across games we experiment with the hyperparameter $p^+ \in [0, 1]$ in 0.1 increment, and the value for LM finetuning frequency $k \in [2k, 5k, 10k, 20k]$. The results tabled are estimated from 5 runs.

6 Results

We follow the questions enumerated in §5 to analyze the effect of in-game learning of language models for action recommendations.

6.1 Effect on Performance

To understand the effect on performance with LM-in-the-Loop, we follow the experimental setup in §5.3 to evaluate the Jericho benchmark. Table 1 compares the different methods detailed in §4.1 with reproduced norm score of CALM as the baseline. We see that categorizing the transitions using state features (**SF**) scored the highest in all tasks, suggesting that LM-in-the-Loop enables improved performance. The improvement in the average norm score was approximately 4% over the baseline. , which translates to about $\approx 53\%$ more average improvement over the scores obtained by the baseline model. We refer to Appendix A for the learning graph for individual games for 5 seeds.

However, the improvement with SF is, in a way, a loose upper bound to in-game learning with LM-in-the-Loop, as special techniques to reweight the transitions (UT^{LA}, and UT^{EA}), or reward-based categorization RT only improved the avg. norm score by $\approx 0.6\%$. On the other hand, the avg. norm

Games	**CALM**$_1$	UT$_2$	UT$_3^{\text{LA}}$	UT$^{4\,\text{EA}}$	RT$_5$	**SF**$_6$	$\Delta(\%)_{(6-1)}$	**Max Score**
Zork1	$30.7_{[4.8]}$	$32.6_{[4.4]}$	$30.4_{[8.5]}$	$35.6_{[5.7]}$	$30.7_{[3.8]}$	$\mathbf{38.0}_{[1.7]}$	23%	350
Inhumane	$24.8_{[2.7]}$	$21.9_{[5.24]}$	$28.9_{[11]}$	$27.3_{[3.1]}$	$29.1_{[12.7]}$	$\mathbf{43.4}_{[3.8]}$	75%	90
Detective	$\mathbf{290.9}_{[2.7]}$	$288.5_{[1.5]}$	$289.3_{[0.2]}$	$288.3_{[1.3]}$	$285.1_{[5.6]}$	$288.5_{[1.5]}$	0%	360
Zork3	$0.3_{[0.09]}$	$0.3_{[0.14]}$	$0.4_{[0.1]}$	$0.6_{[0.1]}$	$0.6_{[0.1]}$	$\mathbf{0.7}_{[0.2]}$	133%	7
Omniquest	$6.7_{[0.3]}$	$6.0_{[0.6]}$	$6.6_{[0.9]}$	$6.6_{[1]}$	$6.0_{[0.79]}$	$\mathbf{7.8}_{[1.7]}$	16%	50
Library	$11.2_{[1.3]}$	$9.3_{[1.1]}$	$9.5_{[1]}$	$10.3_{[0.2]}$	$10.3_{[1.8]}$	$\mathbf{12.1}_{[0.7]}$	8%	30
Balances	$9.3_{[0.2]}$	$9.6_{[0.1]}$	$9.6_{[0.2]}$	$9.5_{[0.2]}$	$9.7_{[0.2]}$	$\mathbf{9.7}_{[0.1]}$	4%	51
Ludicorp	$10.4_{[0.7]}$	$11.4_{[2.6]}$	$12.5_{[1.1]}$	$11.9_{[2.6]}$	$11.3_{[3.1]}$	$\mathbf{15.1}_{[0.8]}$	45%	150
Dragon	$0.1_{[0.06]}$	$0.1_{[0.1]}$	$0.3_{[0.3]}$	$0.3_{[0.3]}$	$0.1_{[0.12]}$	$\mathbf{0.3}_{[0.2]}$	200%	25
Ztuu	$3.8_{[0.18]}$	$4.4_{[0.0]}$	$4.5_{[0.2]}$	$4.4_{[0.1]}$	$4.3_{[0.1]}$	$\mathbf{4.5}_{[0.1]}$	18%	100
Norm Score	20.1%	19.1%	20.6%	20.9%	20.7 %	**24.0**%	52.37%	100%

Table 1: From the results, it can be consistently seen that LM-in-the-Loop provides a performance improvement over CALM. Especially, categorizing the transitions with state features (SF) scored the highest with $\sim 53\%$ improvement over the scores obtained by the baseline model.

score with Uncategorized Transitions (UT) dropped to 19.2% which is $\sim$ 1% below the baseline performance. The difference in performance between UT, and SF with the baseline suggests that LM-in-the-loop for action recommendation is helpful but requires careful selection of transitions for training the language model.

In Figure 3, we compare the % of steps in-game learning methods took on average to achieve $k\%$ of the CALM model's best performance across the games. We see that LM-in-the-Loop techniques enabled at least 2$\times$ on average acceleration in convergence. Although alternatives like reward-based categorization and reweighted techniques only provided meager improvements over the baseline (Table 1), they still show accelerated coverage with 40% to reach the baseline score.

6.2 Effect of Weight Adjusted LM Loss

Categorization of transitions, although possible in most games, often requires game-specific functions to identify what a good and a bad transition is. However, a generalized technique would be to use a notion of the usefulness of transitions that do not require game-specific mechanisms. We explore reweighted cross-entropy loss as in Equation 4 with variations of the $h(\cdot)$ functions from being uniformly distributed as 1 over $(o, a) \in \mathcal{O} \times \mathcal{A}$ to using advantage function with two variations as in Equation 6 and Equation 5. While UT uses vanilla cross-entropy loss to train the LM on transitions sampled from buffer $\mathcal{D}$, UT$^{\text{EA}}$ and UT$^{\text{LA}}$ adjusts the experience according to the advantage, $A(o, a)$, of the actions chosen in those observations.

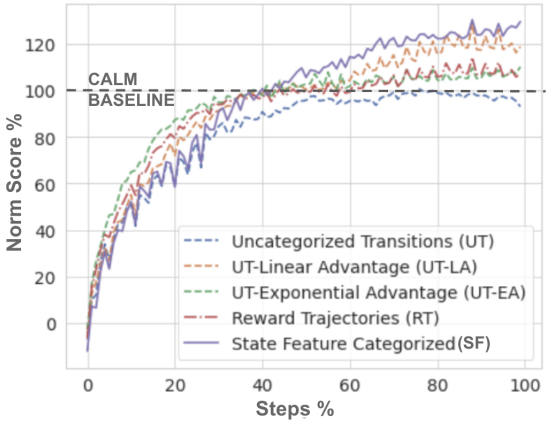


Figure 3: We see that LM-in-the-Loop techniques only need half of the steps to achieve the best of CALM. Using state feature-based categorization (SF) achieves better acceleration and performance over the rest.

We use causal language modeling to train the GPT-2 LM to discourage the LM from generating a helpful action in a state and discourage the not useful. As $A(o, a) \in [-\infty, +\infty]$, it is important to understand how it affects the language model. A negative advantage for a' in o' should discourage the LM from suggesting a' in o'. UT$^{\text{EA}}$ re-scales the LM-loss with $h(\cdot) \in [0, 1)$, while UT$^{\text{LA}}$ works

similar to *Unlikelihood* training as proposed in Welleck et al. (2019) by maintaining the same scale as $A(o, a)$. However, from the results, we see that the differences in reweighting did not tangibly affect the performance as seen in Table 1 (Columns UT^{EA} and UT^{LA}). However, there is an accelerated convergence as shown in the Figure 3. Even with 40% of interactions, it reached the baseline approach-CALM performance.

6.3 Emphasis on Human Annotations

CALM model—the baseline— uses all of the $\sim 223K$ transitions in the ClubFloyd dataset to adapt the GPT-2 model for action recommendation. However, using in-game transitions for LM-in-the-loop training provides the LM with game-specific information. The requirement for adapting GPT-2 with human-annotated transitions should be minimal. The existing approach shows that performance decreased significantly when adaptation was done with 10% of the ClubFloyd dataset. The reproduced results of CALM with 10% of adaptation data show the average norm score as 18.5% across the games in Table 2. Using State features (**SF**) with 10% of the adaptation date achieved an average norm score of 21.8%, more than even using 100% of the adaptation data with CALM. Although there was a small decline in the performance of the detective game, it was insignificant because it was still within the standard error. These results suggest empirically that we can reduce the burden of collecting human-played or human-annotated data by doing in-game learning.

Games	**CALM**$_1$ **100%**	**CALM** $_2$ **10%**	**SF**$_3$ **10%**	Δ(%) $_{(3-2)}$
Zork1	$30.7_{[4.8]}$	$29_{[3.4]}$	$\mathbf{35.1}_{[2.3]}$	21%
Inhumane	$24.8_{[2.7]}$	$15.7_{[14.7]}$	$\mathbf{27.5}_{[6.8]}$	75%
Detective	$290.9_{[2.7]}$	$289.5_{[0.2]}$	$\mathbf{289.6}_{[0.2]}$	0%
Zork3	$0.3_{[0.09]}$	$0.6_{[0]}$	$\mathbf{0.7}_{[0.3]}$	16%
Omniquest	$6.7_{[0.3]}$	$5.9_{[0.8]}$	$\mathbf{6.0}_{[1]}$	1%
Library	$11.2_{[1.3]}$	$\mathbf{10.5}_{[1.5]}$	$10.2_{[1.8]}$	(2)%
Balances	$9.3_{[0.2]}$	$6.6_{[3.5]}$	$\mathbf{8.6}_{[1.6]}$	30%
Ludicorp	$10.4_{[0.7]}$	$10.2_{[0.4]}$	$\mathbf{13.7}_{[0.4]}$	34%
Dragon	$0.1_{[0.06]}$	$0.1_{[0.06]}$	$\mathbf{0.3}_{[0.2]}$	200%
Ztuu	$3.8_{[0.18]}$	$3.6_{[0.1]}$	$\mathbf{4.1}_{[0.1]}$	13%
Norm	20.1%	18.5%	**21.8** %	39.0%

Table 2: Using State Features (SF) achieved an average norm score of 21.8% with 10%, which was more than even with CALM using 100% baseline.

7 Conclusion

In this work, we proposed frameworks for selecting in-game transitions to adjust the LLM to reduce the reliance on human-annotated gameplays while enhancing performance and convergence. We used various sampling strategies to adapt an LM in an online setting by utilizing a buffer to store in-game transitions. The results indicate that categorizing the transitions using state features yielded the best performance across all tasks, demonstrating the effectiveness of LM-in-the-Loop. Furthermore, in-game training accelerates the convergence in most games. Moreover, our findings suggest that LM-in-the-loop training can alleviate the burden of collecting human game plays. In conclusion, adapting a language model using in-game trajectories showed improved performance, faster convergence, and more sample efficiency.

Limitations

The paper analyzes the possibility and challenges in LM-in-the-Loop training of the GPT-2 model for action recommendation in text-based games. The claims in the work can be further supported with experiments on different LLMs. Similarly, the generalization experiments could have added more support to the lack of evidence with additional games. However, these are compute-intensive experiments, and the claims are largely made in consideration of the limitations in the setup.

References

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. Do as i can, not as i say: Grounding language in robotic affordances, 2022. URL https://arxiv.org/abs/2204.01691.

Prithviraj Ammanabrolu and Matthew Hausknecht. Graph constrained reinforcement learning for natural language action spaces. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=B1x6w0EtwH.

Emily M. Bender and Alexander Koller. Climbing towards NLU: On meaning, form, and understanding in the age of data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 5185–5198, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.463. URL https://aclanthology.org/2020.acl-main.463.

Lukas Biewald. Experiment tracking with weights and biases, 2020. URL https://www.wandb.com/. Software available from wandb.com.

Yonatan Bisk, Ari Holtzman, Jesse Thomason, Jacob Andreas, Yoshua Bengio, Joyce Chai, Mirella Lapata, Angeliki Lazaridou, Jonathan May, Aleksandr Nisnevich, Nicolas Pinto, and Joseph Turian. Experience grounds language. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 8718–8735, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.703. URL https://aclanthology.org/2020.emnlp-main.703.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D 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 Ziegler, Jeffrey Wu, Clemens Winter, Chris 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. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=a7APmM4B9d.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*, 2014.

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, 2018. URL http://arxiv.org/abs/1806.11532.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran,

and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pp. 4171–4186. Association for Computational Linguistics, 2019a. doi: 10.18653/v1/n19-1423. URL https://doi.org/10.18653/v1/n19-1423.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pp. 4171–4186. Association for Computational Linguistics, 2019b. doi: 10.18653/v1/n19-1423. URL https://doi.org/10.18653/v1/n19-1423.

Steven Y Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. A survey of data augmentation approaches for nlp. *arXiv preprint arXiv:2105.03075*, 2021.

Matthew Hausknecht, Prithviraj Ammanabrolu, Marc-Alexandre Côté, and Xingdi Yuan. Interactive fiction games: A colossal adventure. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7903–7910, Apr. 2020. doi: 10.1609/aaai.v34i05.6297. URL https://ojs.aaai.org/index.php/AAAI/article/view/6297.

Ji He, Jianshu Chen, Xiaodong He, Jianfeng Gao, Lihong Li, Li Deng, and Mari Ostendorf. Deep reinforcement learning with a natural language action space. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1621–1630, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1153. URL https://aclanthology.org/P16-1153.

Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 1273–1286. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/099fe6b0b444c23836c4a5d07346082b-Paper.pdf.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pp. 427–431, Valencia, Spain, April 2017. Association for Computational Linguistics. URL https://aclanthology.org/E17-2068.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. UNIFIEDQA: Crossing format boundaries with a single QA system. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 1896–1907, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.171. URL https://aclanthology.org/2020.findings-emnlp.171.

Brenden M. Lake and Gregory L. Murphy. Word meaning in minds and machines. *Psychological review*, 2021.

Shuang Li, Xavier Puig, Chris Paxton, Yilun Du, Clinton Wang, Linxi Fan, Tao Chen, De-An Huang, Ekin Akyürek, Anima Anandkumar, Jacob Andreas, Igor Mordatch, Antonio Torralba, and Yuke Zhu. Pre-trained language models for interactive decision-making. *arXiv*, 2022.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P11-1015.

James L McClelland, Felix Hill, Maja Rudolph, Jason Baldridge, and Hinrich Schütze. Placing language in an integrated understanding system: Next steps toward human-level performance in neural language models. *Proceedings of the National Academy of Sciences*, 117(42):25966–25974, 2020.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Gray, 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. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=TG8KACxEON.

Emilio Parisotto, H. Francis Song, Jack W. Rae, Razvan Pascanu, Çaglar Gülçehre, Siddhant M. Jayakumar, Max Jaderberg, Raphael Lopez Kaufman, Aidan Clark, Seb Noury, Matthew M. Botvinick, Nicolas Heess, and Raia Hadsell. Stabilizing transformers for reinforcement learning. *CoRR*, abs/1910.06764, 2019. URL http://arxiv.org/abs/1910.06764.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training, 2018a.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners, 2018b. URL https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264. URL https://aclanthology.org/D16-1264.

Machel Reid, Yutaro Yamada, and Shixiang Shane Gu. Can wikipedia help offline reinforcement learning? *CoRR*, abs/2201.12122, 2022. URL https://arxiv.org/abs/2201.12122.

Ishika Singh, Gargi Singh, and Ashutosh Modi. Pre-trained language models as prior knowledge for playing text-based games. *CoRR*, abs/2107.08408, 2021. URL https://arxiv.org/abs/2107.08408.

Denis Tarasov, Vladislav Kurenkov, and Sergey Kolesnikov. Prompts and pre-trained language models for offline reinforcement learning. In *ICLR 2022 Workshop on Generalizable Policy Learning in Physical World*, 2022. URL https://openreview.net/forum?id=Spf4TE6NkWq.

Jens Tuyls, Shunyu Yao, Sham M. Kakade, and Karthik R Narasimhan. Multi-stage episodic control for strategic exploration in text games. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=Ek7PSN7Y77z.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. Neural text generation with unlikelihood training. *arXiv preprint arXiv:1908.04319*, 2019.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online, October 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-demos.6. URL https://aclanthology.org/2020.emnlp-demos.6.

Yuhuai Wu, Felix Li, and Percy Liang. Insights into pre-training via simpler synthetic tasks. *arXiv preprint arXiv:2206.10139*, 2022.

Yunqiu Xu, Ling Chen, Meng Fang, Yang Wang, and Chengqi Zhang. Deep reinforcement learning with transformers for text adventure games. In *2020 IEEE Conference on Games (CoG)*, pp. 65–72, 2020. doi: 10.1109/CoG47356.2020.9231622.

Shunyu Yao, Rohan Rao, Matthew Hausknecht, and Karthik Narasimhan. Keep CALM and explore: Language models for action generation in text-based games. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 8736–8754, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.704. URL https://aclanthology.org/2020.emnlp-main.704.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper/2015/file/250cf8b51c773f3f8dc8b4be867a9a02-Paper.pdf.

A Appendix

A.1 Language Model and Reinforcement Learning Setup

We use a GPT-2-Base (Radford et al., 2018b) model with 12-layers, 768-hidden units, and 12- attention heads with 117M parameters pre-trained on the WebText corpus. This model's implementation and pre-trained weights are obtained from (Wolf et al., 2020, Huggingface).

We train for 3 epochs on the ClubFloyd dataset following (Yao et al., 2020) to minimize the cross-entropy loss, as shown in Table 3. We use AdamW to optimize the model's weights to minimize the loss, with the learning rate as 2×10^{-6} and Adam epsilon as 1×10^{-9}. We use a linear schedule with a warmup of 0.1 for the learning rate. Finally, we clip gradients with a maximum gradient norm of 1. Following Yao et al. (2020)'s finetuning process, we exclude using Jericho-related transcripts by setting the flag as 1. We used random seeds to select the dataset to avoid bias when selecting data for the LM training.

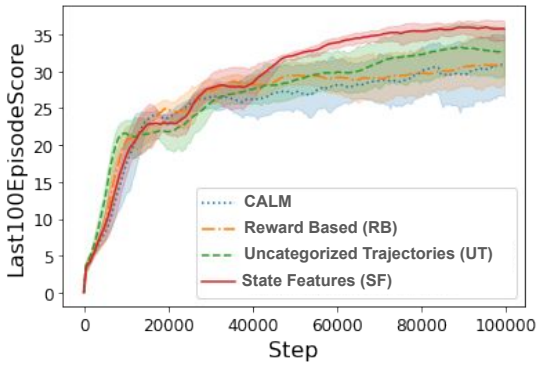
We train on 10 interactive fiction games from the Jericho benchmark (Hausknecht et al., 2020). The states are observations concatenated with items in the player's possession and their current location description provided by the game engine using commands inventory and look. A single game episode runs for 100 environment steps at max or gets terminated before the game is over or won. We use the `look` and `inventory` commands to add location and inventory descriptions to observations, following Hausknecht et al. (2020).

Model	Metric	Final Score
100%	Train Loss	1.49
	Val Loss	2.65
	Train Acc	0.30
	Val Acc	0.14
10%	Train Loss	1.42
	Val Loss	3.04
	Train Acc	0.30
	Val Acc	0.09

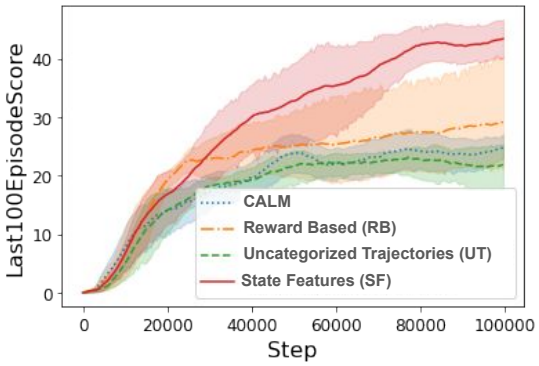
Table 3: Pre-trained GPT-2 Language Model training details on different data percentage variants trained for 3 epochs.

We train DRRN asynchronously on 8 parallel instances of the game environment for $100,000$ steps for each game. At each step, the Q-value is estimated using the DRRN agent, and the action is selected based on the soft-exploration policy. Action's admissibility is predicted based on the textual response of the game. Then, inadmissible are filtered out using a FastText model (Joulin et al., 2017). The agent is optimized using Adam optimizer with a 10^{-5} learning rate. We sample transitions of batch size 64 from priority buffer with a priority fraction of 0.5. The discount factor in determining the importance of the future reward is 0.9. The size of the embedding dimension is 128, and the hidden dimension is 128. Finally, the gradient is clipped with a maximum gradient norm of 5. We train 5 separate runs for each game and report the average score. We use the average of the last 100 episode scores to calculate the final score.

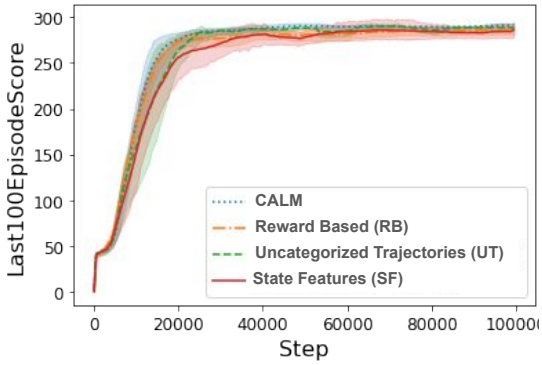
A.2 Learning Plots



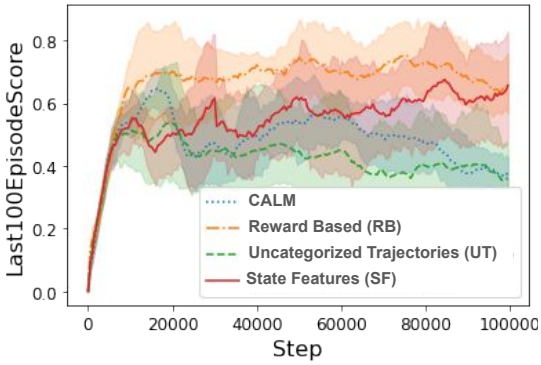
(a) Zork 1



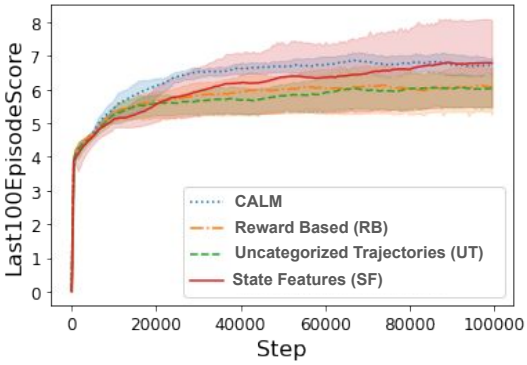
(b) Inhumane



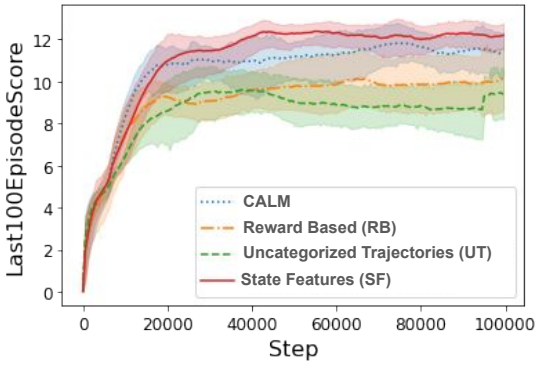
(c) Detective



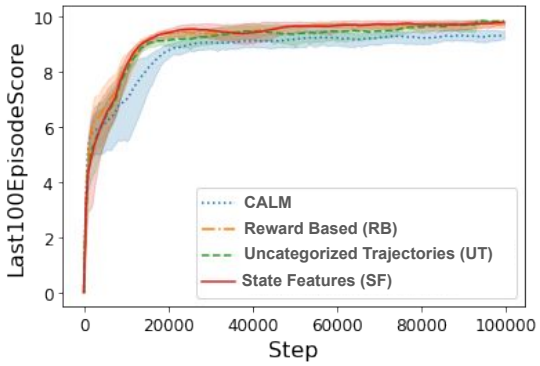
(d) Zork3



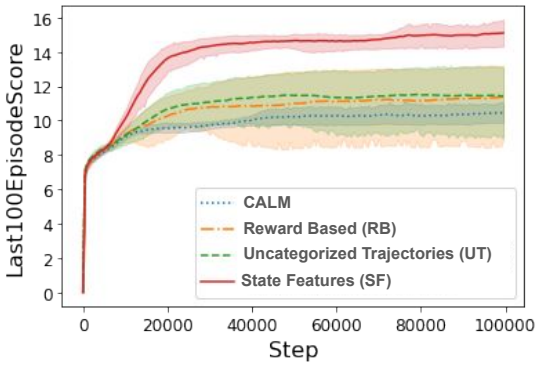
(e) Omniquest



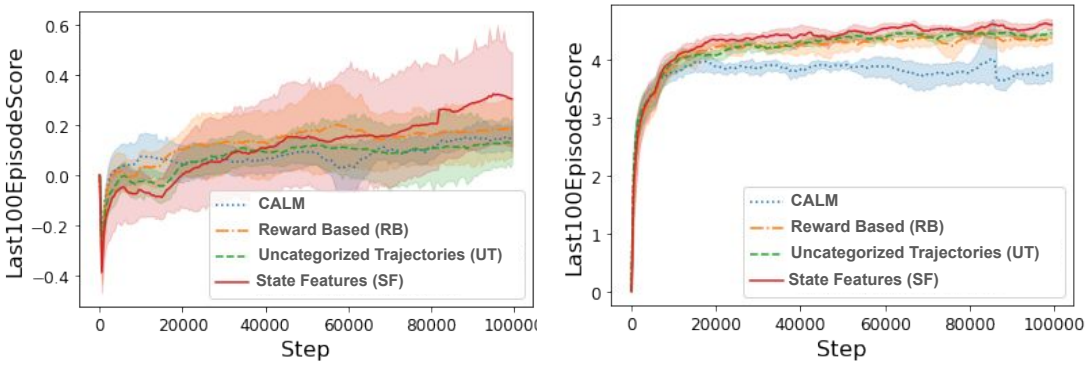
(f) Library



(g) Balances



(h) Ludicorp



(i) Dragon (j) Ztuu

Figure 4: Comparison of learning dynamics of the different LM-in-the-Loop techniques with the baseline CALM agent across the selected 10 games in Jericho for 5 seeds.

A.3 Software Details

We used PyTorch for the code implementation and Huggingface to load pre-trained language models. We used Weights & Biases (Biewald, 2020) for experiment tracking and visualizations to develop insights for this paper. Finally, the seaborn package is used to generate plots.