A Minimal Approach for Natural Language Action Space in Text-based Games

Dongwon Kelvin Ryu[♠] **Meng Fang**[♡] **Gholamreza Haffari**[♠]
Shirui Pan[♢] **Ehsan Shareghi**[♠ ♣]
[♠] Department of Data Science & AI, Monash University
[♡] University of Liverpool [♢] Griffith University
[♣] Language Technology Lab, University of Cambridge
firstname.lastname@monash.edu Meng.Fang@liverpool.ac.uk
s.pan@griffith.edu.au

Abstract

Text-based games (TGs) are language-based interactive environments for reinforcement learning. While language models (LMs) and knowledge graphs (KGs) are commonly used for handling large action space in TGs, it is unclear whether these techniques are necessary or overused. In this paper, we revisit the challenge of exploring the action space in TGs and propose ϵ-admissible exploration, a minimal approach of utilizing admissible actions, for training phase. Additionally, we present a text-based actor-critic (TAC) agent that produces textual commands for game, solely from game observations, without requiring any KG or LM. Our method, on average across 10 games from Jericho, outperforms strong baselines and state-of-the-art agents that use LM and KG. Our approach highlights that a much lighter model design, with a fresh perspective on utilizing the information within the environments, suffices for an effective exploration of exponentially large action spaces. [1]

1 Introduction

An intelligent agent that communicates in natural language space has been a long goal of artificial intelligence (Fang et al., 2017). Text-based games (TGs) best suit this goal, since they allow the agent to *read the textual description of the world* and *write the textual command to the world* (Hausknecht et al., 2020; Côté et al., 2018). In TGs, the agent should perform natural language understanding (NLU), sequential reasoning and natural language generation (NLG) to generate a series of actions to accomplish the goal of the game, i.e. adventure or puzzle (Hausknecht et al., 2020). The language perspective of TGs foists environments partially observable and action space combinatorially large, making the task challenging. Since TGs alert the player how much the game has proceeded with the game score, reinforcement learning (RL) naturally lends itself as a suitable framework.

Due to its language action space, an RL agent in TGs typically deals with a combinatorially large action space, motiving various design choices to account for it. As two seminal works in this space, Yao et al. (2020) trained a language model (LM) to produce admissible actions[2] for the given textual observation and then used, under the predicted action list, Deep Reinforcement Relevance Network to estimate the Q value. As an alternative, Ammanabrolu and Hausknecht (2020) constructs a knowledge graph (KG) to prune down action space while learning the policy distribution through actor-critic (AC) method and supervision signal from the admissible actions. Both paradigms leverage admissible actions at different stages at the cost of imposing additional modules and increasing model complexity.

In this paper, we take a fresh perspective on leveraging the information available in the TG environment to explore the action space without relying on LMs or KGs. We propose a minimal form of utilizing admissibility of actions to constrain the action space during training while allowing the agent to act independently to access the admissible actions during testing. More concretely, our proposed training strategy, ϵ-admissible exploration, leverages the admissible actions via random sampling during training to acquire diverse and useful data from the environment. Then, our developed text-based actor-critic (TAC) agent learns the policy distribution without any action space constraints. It is noteworthy that our much lighter proposal is under the same condition as other aforementioned methods since all the prior works use admissible actions in training the LM or the agent.

Our empirical findings, in Jericho, illustrate that

[1] The code is available at https://github.com/ktr0921/tac

[2] Admissible actions are grounded actions that are guaranteed to change the world state produced by the environment (Hausknecht et al., 2020; Côté et al., 2018).

Proceedings of the 27th Conference on Computational Natural Language Learning (CoNLL), pages 138–154
December 6–7, 2023.

TAC with ϵ-admissible exploration has better or on-par performance in comparison with the state-of-the-art agents that use an LM or KG. Through experiments, we observed that while previous methods have their action selections largely dependent on the quality of the LM or KG, sampling admissible actions helps with the action selection and results in acquiring diverse experiences during exploration. While showing a significant success on TGs, we hope our approach encourages alternative perspectives on leveraging action admissibility in other domains of applications where the action space is discrete and combinatorially large.

2 Basic Definitions

Text-based Games. TGs are game simulation environments that take natural language commands and return textual description of the world. They have received significant attention in both NLP and RL communities in recent years. Côté et al. (2018) introduced TextWorld, a TG framework that automatically generates textual observation through knowledge base in a game engine. It has several hyper-parameters to control the variety and difficulty of the game. Hausknecht et al. (2020) released Jericho, an open-sourced interface for human-made TGs, which has become the de-facto testbed for developments in TG.

Admissible Action. A list of natural language actions that are guaranteed to be understood by the game engine and change the environment in TGs are called Admissible Actions. The term was introduced in TextWorld while a similar concept also exists in Jericho under a different name, valid actions. Hausknecht et al. (2020) proposed an algorithm that detects a set of admissible actions provided by Jericho suite by constructing a set of natural language actions from every template with detectable objects for a given observation and running them through the game engine to return those actions that changed the world object tree.

Template-based Action Space. Natural language actions are built with template ($\mathbb{T}$) and object ($\mathbb{O}$) from template-based action space. Each template takes at most two objects. For instance, a template-object pair (`take OBJ from OBJ`, `egg`, `fridge`) produces a natural language action `take egg from fridge` while (`west`,`-`,`-`) produces `west`.

Partially Observable Markov Decision Process. TG environments can be formalized as Partially Observable Markov Decision Processes (POMDPs). A POMDP is defined as a 7-tuple, $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{O}, \mathcal{P}_o, \mathcal{R}, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ are a set of state and action, and $\mathcal{P}$ is the state transition probability that maps state-action pair to the next state, $\Pr(s_{t+1}|s_t, a_t)$. $\mathcal{O}$ is a set of observation that depends on the current state via an emission probability, $\mathcal{P}_o \equiv \Pr(o_t|s_t)$. $\mathcal{R}$ is an immediate reward signal held between the state and the next state, $r(s_t, s_{t+1})$, and γ is the discount factor. The action selection rule is referred to as the policy $\pi(a|o)$, in which the optimal policy acquires the maximum rewards in the shortest move.

TG Environment as POMDP. Three textual observations are acquired from the engine, game feedback o_{game}, room description o_{look}, and inventory description o_{inv}. The game feedback is dependent on the previous action, $\Pr(o_{\text{game},t}|s_t, a_{t-1})$, while room and inventory descriptions are not, $\Pr(o_{\text{look},t}|s_t)$ and $\Pr(o_{\text{inv},t}|s_t)$. Inadmissible actions do not influence the world state, room and inventory descriptions but change the game feedback changes. Each action is sampled sequentially from template-based action space. For template, we directly sample from observation $\pi(a_{\mathbb{T}}|o)$ while an object policy is sequentially produced, $\pi(a_{\mathbb{O}}|o, \hat{a})$, where $\hat{a}$ is previously sampled template-object pair. The agent ought to find the optimal policy that maximizes the expected discounted sum of rewards, or the return, $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$.

Traditional Reinforcement Learning. There are three traditional algorithms in RL, Q-learning (QL), policy gradient (PG) and actor-critic (AC). QL estimates the return for a given state-action pair, or Q value, $Q(s_t, a_t) = \mathbb{E}[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|s_t, a_t]$, then selects the action of the highest Q value. However, this requires the action space to be countably finite. To remedy this, PG directly learns the policy distribution from the environment such that it maximizes the total return through Monte-Carlo (MC) sampling. AC combines QL and PG, where it removes MC in PG and updates the parameters per each step with estimated Q value using QL. This eliminates the high variance of MC as an exchange of a relatively small bias from QL.

3 Related Work on TG Agents in RL

We provide a brief overview of widely known TG agents relevant to the work presented in this paper. We empirically compare these in the Section 5.1.

Contextual Action LM (CALM)-DRRN (Yao et al., 2020) uses an LM (CALM) to produce a

set of actions for a given textual observation from the TGs. It is trained to map a set of textual observations to the admissible actions through causal language modeling. Then, Deep Reinforcement Relevance Network (DRRN) agent was trained on the action candidates from CALM. DRRN follows QL, estimating the Q value per observation-action pair. As a result, CALM removes the need for the ground truth while training DRRN.[3]

Knowledge Graph Advantage Actor Critic (KG-A2C) (Ammanabrolu and Hausknecht, 2020) uses the AC method to sequentially sample templates and objects, and KGs for long-term memory and action pruning. Throughout the gameplay, KG-A2C organizes knowledge triples from textual observation using Stanford OpenIE (Angeli et al., 2015) to construct a KG. Then, the KG is used to build state representation along with encoded game observations and constrain object space with only the entities that the agent can reach within KG, i.e. immediate neighbours. They used admissible actions in the cross entropy supervised loss.

KG-A2C Inspired Agents. Xu et al. (2020) proposed SHA-KG that uses stacked hierarchical attention on KG. Graph attention network (GAT) was applied to sample sub-graphs of KG to enrich the state representation on top of KG-A2C. Ammanabrolu et al. (2020) used techniques inspired by Question Answering (QA) with LM to construct the KG. They introduced Q*BERT which uses ALBERT (Lan et al., 2020) fine-tuned on a dataset specific to TGs to perform QA and extract information from textual observations of the game, i.e. "Where is my current location?". This improved the quality of KG, and therefore, constituted better state representation. Ryu et al. (2022) proposed an exploration technique that injects commonsense directly into action selection. They used log-likelihood score from commonsense transformer (Bosselut et al., 2019) to re-rank actions. Peng et al. (2021) investigated explainable generative agent (HEX-RL) and applied hierarchical graph attention to symbolic KG-based state representations. This was to leverage the graph representation based on its significance in action selection. They also employed intrinsic reward signal towards the expansion of KG to motivate the agent for exploration (HEX-RL-IM) (Peng et al., 2021).

[3]It is noteworthy, orthogonal to the focus of our work, the recently proposed eXploit-Then-eXplore (Tuyls et al., 2022) uses LM and admissible actions to resolve another challenge, exploration-exploitation dilemma in TGs.

All the aforementioned methods utilize admissible actions in training the LM or agent. Our proposed method, introduced shortly (§4), uses admissible actions as action constraints during training without relying on KG or LM.

4 Text-based Actor Critic (TAC)

Our agent, Text-based Actor Critic (TAC), follows the Actor-Critic method with template-object decoder. We provide an overview of the system in Figure 1 and a detailed description in below. We follow the notation introduced earlier in Section 2.

Encoder. Our design consists of text and state encoders. Text encoder is a single shared bi-directional GRU with different initial hidden state for different input text, ($o_{\text{game}}, o_{\text{look}}, o_{\text{inv}}, a_N$). The state representation only takes encoded textual observations while the natural language action a_N is encoded to be used by the critic (introduced shortly). State encoder embeds game scores into a high dimensional vector and adds it to the encoded observation. This is then, passed through a feed-forward neural network, mapping an instance of observation to state representation without the history of the past information.

Actor. The Actor-Critic design is used for our RL component. We describe our generative actor first. Our actor network maps from state representation to action representation. Then, the action representation is decoded by GRU-based template and object decoders (Ammanabrolu and Hausknecht, 2020). Template decoder takes action representation and produces the template distribution and the context vector. Object decoder takes action representation, semi-completed natural language action and the context from template decoder to produce object distribution sequentially.

Critic. Similar to (Haarnoja et al., 2018), we employed two types of critics for practical purpose, state critic for state value function and state-action critic for state-action value function. Both critics take the state representation as input, but state-action critic takes encoded natural language action as an additional input. The textual command produced by the decoder is encoded with text encoder and is passed through state-action critic to predict state-action value, or Q value, for a given command. A more detailed diagram for Actor and Critic is in Appendix D. To smooth the training, we introduced target state critic as an exponentially moving average of state critic (Mnih et al., 2015). Also, the

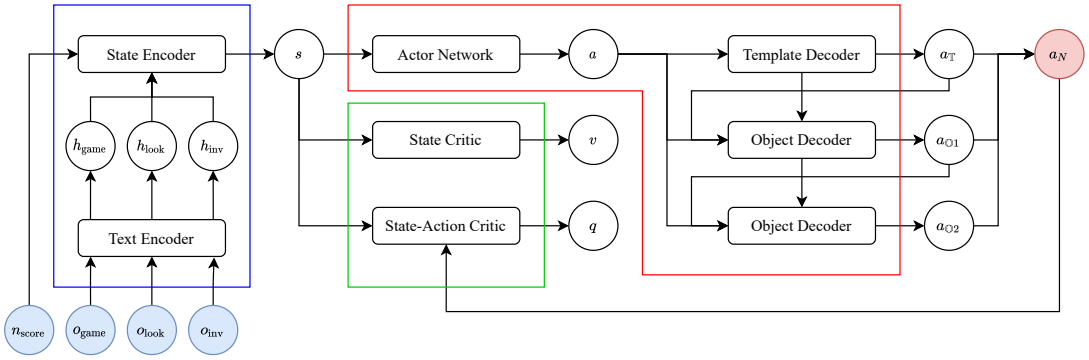

Figure 1: Text-based Actor-Critic (TAC); A blue circle is the input to the encoder, ($n_{\text{score}}, o_{\text{game}}, o_{\text{look}}, o_{\text{inv}}$) representing (game score, game feedback, room description, inventory), while a red circle is the output from actor, a_N representing natural language action. Blue, red and green boxes indicate encoder, actor and critic, respectively.

two state-action critics are independently updated to mitigate positive bias in the policy improvement (Fujimoto et al., 2018). We used the minimum of the two enhanced critic networks outputs as our estimated state-action value function.

Objective Function. Our objective functions are largely divided into two, RL and SL. RL objectives are for reward maximization $\mathcal{L}_{\text{R}}$, state value prediction $\mathcal{L}_{\text{V}}$, and state-action value prediction $\mathcal{L}_{\text{Q}}$. We overload the notation of θ: for instance, $V_\theta(o)$ signifies parameters from the encoder to the critic, and $\pi_\theta(a|o)$ from the encoder to the actor. Reward maximization is done as follows,

$$\mathcal{L}_{\text{R}} = -\mathbb{E}\left[A(o,a)\nabla_\theta \ln \pi_\theta\left(a|o\right)\right], \quad (1)$$

$$A(o,a) = Q_\theta(o,a) - V_\theta(o), \quad (2)$$

where $A(o,a)$ is the normalized advantage function with no gradient flow.

$$\mathcal{L}_{\text{V}} = \mathbb{E}\left[\nabla_\theta\left(V_\theta(o) - \left(r + \gamma V_{\bar{\theta}}(o')\right)\right)\right], \quad (3)$$

$$\mathcal{L}_{\text{Q}} = \mathbb{E}\left[\nabla_\theta\left(Q_\theta(o,a) - \left(r + \gamma V_{\bar{\theta}}(o')\right)\right)\right], \quad (4)$$

where o' is observation in the next time step and $\bar{\theta}$ signifies the parameters containing the target state critic, updated as moving average with τ,

$$\bar{\theta}_v = \tau\theta_v + (1-\tau)\bar{\theta}_v. \quad (5)$$

Our SL updates the networks to produce valid templates and valid objects,

$$\mathcal{L}_{\mathbb{T}} = \frac{1}{|\mathbb{T}|}\sum_{a_{\mathbb{T}}\in\mathbb{T}}\left(y_{a_{\mathbb{T}}}\ln\left(\pi_\theta(a_{\mathbb{T}}|o)\right) + \left(1-y_{a_{\mathbb{T}}}\right)\left(1-\ln\left(\pi_\theta(a_{\mathbb{T}}|o)\right)\right)\right), \quad (6)$$

$$\mathcal{L}_{\mathbb{O}} = \frac{1}{|\mathbb{O}|}\sum_{a_{\mathbb{O}}\in\mathbb{O}}\left(y_{a_{\mathbb{O}}}\ln\left(\pi_\theta(a_{\mathbb{O}}|o,\hat{a})\right) + \left(1-y_{a_{\mathbb{O}}}\right)\left(1-\ln\left(\pi_\theta(a_{\mathbb{O}}|o,\hat{a})\right)\right)\right), \quad (7)$$

$$y_{a_{\mathbb{T}}} = \begin{cases} 1 & a_{\mathbb{T}} \in \mathbb{T}_a \\ 0 & \text{otherwise} \end{cases} \qquad y_{a_{\mathbb{O}}} = \begin{cases} 1 & a_{\mathbb{O}} \in \mathbb{O}_a \\ 0 & \text{otherwise} \end{cases}$$

where $\mathcal{L}_{\mathbb{T}}$ and $\mathcal{L}_{\mathbb{O}}$ are the cross entropy losses over the templates ($\mathbb{T}$) and objects ($\mathbb{O}$). Template and object are defined as $a_{\mathbb{T}}$ and $a_{\mathbb{O}}$, while $\hat{a}$ is the action constructed by previously sampled template and object. Positive samples, $y_{a_{\mathbb{T}}}$ and $y_{a_{\mathbb{O}}}$, are only if the corresponding template or object are in the admissible template ($\mathbb{T}_a$) or admissible object ($\mathbb{O}_a$).[4] The final loss function is constructed with λ coefficients to control for trade-offs,

$$\mathcal{L} = \lambda_{\text{R}}\mathcal{L}_{\text{R}} + \lambda_{\text{V}}\mathcal{L}_{\text{V}} + \lambda_{\text{Q}}\mathcal{L}_{\text{Q}} + \lambda_{\mathbb{T}}\mathcal{L}_{\mathbb{T}} + \lambda_{\mathbb{O}}\mathcal{L}_{\mathbb{O}}. \quad (8)$$

Our algorithm is akin to vanilla A2C proposed by Ammanabrolu and Hausknecht (2020) with some changes under our observations. A detailed comparison and qualitative analysis are in Appendix E and F.

ϵ-admissible Exploration. We use a simple exploration technique during training, which samples the next action from admissible actions with ϵ probability threshold. For a given state s, define $\mathcal{A}_a(s) \subseteq \mathcal{A}_N$ as an admissible action subset of all natural language actions set. We sample an action directly from admissible action set under uniform distribution, $a_N \sim \mathcal{U}(\mathcal{A}_a(s))$. Formally, we uniformly sample $p \in [0,1]$ per every step,

$$\beta(a|s) = \begin{cases} \mathcal{U}(\mathcal{A}_a(s)) & p < \epsilon \\ \pi(a|s) & p \geq \epsilon \end{cases} \quad (9)$$

This collects diverse experiences from altering the world with admissible actions. We also tried a variant where the ϵ is selected adaptively given the game score the agent has achieved. However, this variant under-performed the static ϵ. See Appendix I for more details on this and the results.

[4] Eq. 7 is calculated separately for two objects in a single template, where the admissible object space ($\mathbb{O}_a$) is conditioned on the previously sampled template and object.

	LM-BASED	KG-BASED					
Games	CALM-DRRN	KG-A2C	SHA-KG	Q*BERT	HEX-RL	HEX-RL-IM	TAC
BALANCES	9.1	**10.0**	9.8	**10.0**	**10.0**	**10.0**	**10.0** $\pm$ **0.1**
DEEPHOME	1.0	1.0	1.0	1.0	1.0	1.0	**25.4** $\pm$ **3.2**
DETECTIVE	**289.7**	207.9	246.1	274.0	276.7	276.9	272.3 $\pm$ 23.3
LIBRARY	9.0	14.3	10.0	**18.0**	15.9	13.8	**18.0** $\pm$ **1.2**
LUDICORP	10.1	17.8	17.6	**18.0**	14.0	17.6	7.7 $\pm$ 2.5
PENTARI	0.0	50.7	48.2	50.0	34.6	44.7	**53.2** $\pm$ **2.9**
TEMPLE	0.0	7.6	7.9	**8.0**	**8.0**	**8.0**	5.8 $\pm$ 2.3
ZORK1	30.4	34.0	33.6	35.0	29.8	30.2	**46.3** $\pm$ **5.0**
ZORK3	0.5	0.1	0.7	0.1	–	–	**1.6** $\pm$ **1.2**
ZTUU	3.7	5.0	5.0	5.0	5.0	5.1	**33.2** $\pm$ **26.3**
NORMALIZED MEAN	0.1549	0.2475	0.2490	0.2788	0.2722†	0.2834†	**0.3307**

Table 1: Game score comparison over 10 popular game environments in Jericho, with best results highlighted by **boldface**. We only included algorithms that reported the end performance. †HEX-RL and HEX-RL-IM did not report the performance in ZORK3 and are not open-sourced, so the mean average did not account ZORK3.

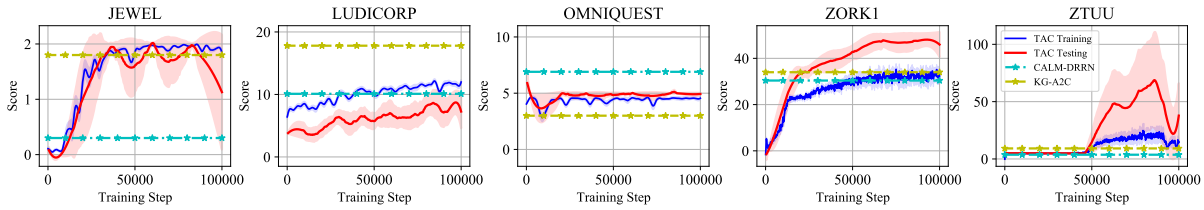

Figure 2: The full learning curve of TAC on five games in Jericho suite. Blue and red plots are training and testing game score while cyan and yellow star marker line signify CALM-DRRN and KG-A2C.

5 Experiments

In this section, we provide a description of our experimental details and discuss the results. We selected a wide variety of agents (introduced in Section 3) utilizing the LM or the KG: CALM-DRRN (Yao et al., 2020) and KG-A2C (Ammanabrolu and Hausknecht, 2020) as baselines, and SHA-KG (Xu et al., 2020), Q*BERT (Ammanabrolu et al., 2020), HEX-RL and HEX-RL-IM (Peng et al., 2021) as state-of-the-art (SotA).

Experimental Setup. Similar to KG-A2C, we train our agent on 32 parallel environments with 5 random seeds. We trained TAC on games of Jericho suite with 100k steps and evaluated with 10 episodes per every 500 training step. During the training, TAC uses uniformly sampled admissible action for a probability of ϵ and during the testing, it follows its policy distribution generated from the game observations. We used prioritized experience replay (PER) as our replay buffer (Schaul et al., 2016). We first fine-tune TAC on ZORK1, then apply the same hyper-parameters for all the games. The details of our hyper-parameters can be found in Appendix A. Our final score is computed as the average of 30 episodic testing game scores. Additionally, our model has a parameter size of less than 2M, allowing us to run the majority of our experiments on CPU (Intel Xeon Gold 6150 2.70 GHz). The full parameter size in ZORK1 and the training time comparison can be found in Appendices B and C.

5.1 Main Results

Table 1 reports the results for baselines, SotAs and TAC on 10 popular Jericho games. TAC attains the new SotA scores in 5 games. Apart from PENTARI, TAC surpasses 4 games with a large margin, where all of the other agents fail to pass the performance bottleneck (DEEPHOME with 1, ZORK1 with 35, ZORK3 with 1, and ZTUU with 5). In DETECTIVE, TAC matches many SotAs, but falls short in LUDICORP and TEMPLE. Nevertheless, TAC achieves the highest mean score over LM or KG-based methods.

On a larger set of 29 games in comparison with the baselines, TAC surpasses CALM-DRRN in *14 out of 29 games* and KG-A2C in *16 out of 29 games* and achieves more than $\sim$ 50% higher score than both CALM-DRRN and KG-A2C with normalized mean score. Per game, in SORCERER, SPIRIT, ZORK3 and ZTUU, TAC achieves at least $\sim$ 200% and at most $\sim$ 400% higher score.. In ACORNCOURT, DEEPHOME and DRAGON, both

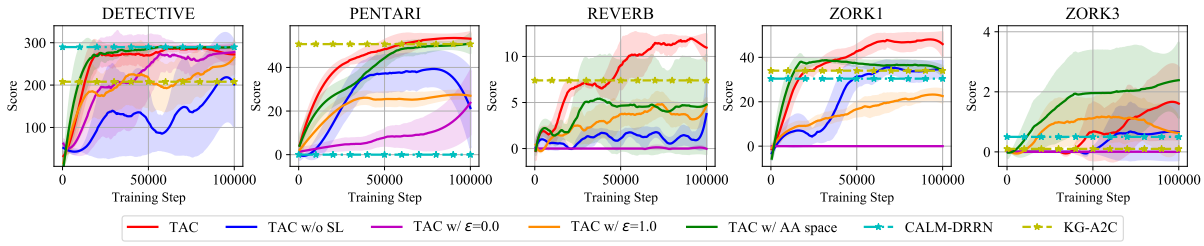

Figure 3: Ablation study on five popular games in Jericho suite. Four different ablation are conducted with SL, $\epsilon = 0.0$, $\epsilon = 1.0$, and with full admissible constraints during training (Admissible Action space). Similar to the previous figure, CALM-DRRN and KG-A2C are added for comparison.

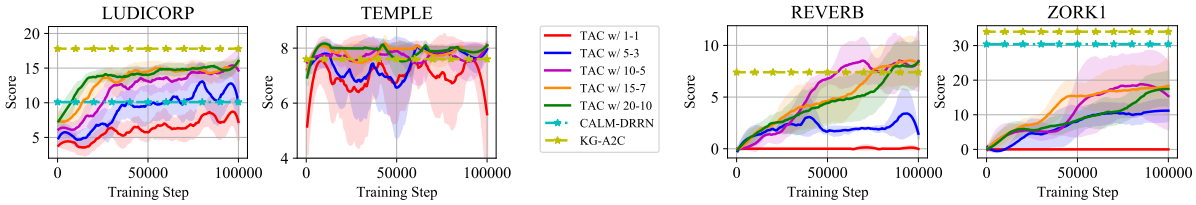

Figure 4: The learning curve of TAC for stronger supervised signals where 5-3 signifies $\lambda_{\mathbb{T}} = 5$ and $\lambda_{\mathbb{O}} = 3$. Left two plots are with $\epsilon = 0.3$ and right two are with $\epsilon = 0$.

CALM-DRRN and KG-A2C fails to achieve any game score (approximately 0), but TAC achieves the score of $+3.4$, $+25.4$ and $+2.81$ For detailed game scores and the full learning curves on 29 games, please refer to Appendix G.

There are a few games that TAC under-performs. We speculate three reasons for this: over-fitting, exploration, and catastrophic forgetting. For instance, as illustrated by the learning curves of TAC in Figure 2, LUDICORP appears to acquire more reward signals during training, but fails to achieve them during testing. We believe this is because the agent is over-fitted to spurious features in specific observations (Song et al., 2020), producing inadmissible actions for a given state that are admissible in other states. On the other hand, TAC in OMNIQUEST cannot achieve a game score more than 5 in both training and testing. This is due to the lack of exploration, where the agent is stuck at certain states because the game score is too far to reach. This, in fact, occurs in ZORK3 and ZTUU for some random seeds, where few seeds in ZORK3 do not achieve any game score while ZTUU achieves 10 or 13 only, resulting in high variance. Finally, catastrophic forgetting (Kirkpatrick et al., 2016) is a common phenomenon in TGs (Hausknecht et al., 2020; Ammanabrolu and Hausknecht, 2020), and this is also observed in JEWEL with TAC.

Training Score vs. Testing Score. Figure 2 shows that the game scores during training and testing in many games are different. There are three interpretations for this: (i) the ϵ-admissible exploration triggers negative rewards since it is uniformly sampling admissible actions. It is often the case that negative reward signal triggers termination of the game, i.e. -10 score in ZORK1, so this results in episodic score during training below testing. (ii) the ϵ-admissible exploration sends the agent to the rarely or never visited state, which is commonly seen in ZTUU. This induces the agent taking useless actions that would not result in rewards since it does not know what to do. (iii) Over-fitting where testing score is lower than training score. This occurs in LUDICORP, where the agent cannot escape certain states with its policy during testing. ϵ-admissible exploration lets the agent escape from these state during training, and therefore, achieves higher game score.

5.2 Ablation

ϵ-Admissible Exploration. To understand how ϵ influences the agent, ablations with two ϵ values, 0.0 and 1.0, on five selective games were conducted. As shown in Figure 3, in the case of $\epsilon = 0.0$, the agent simply cannot acquire reward signals. TAC achieves 0 game score in REVERB, ZORK1 and ZORK3 while it struggles to learn in DETECTIVE and PENTARI. This indicates that the absence of ϵ-admissible exploration results in meaningless explorations until admissible actions are reasonably learned through supervised signals. With $\epsilon = 1.0$, learning becomes unstable

since this is equivalent to no exploitation during training, not capable of observing reward signals that are far from the initial state. Hence, tuned ϵ is important to allow the agent to cover wider range of states (exploration) while acting from its experiences (exploitation).

Supervised Signals. According to the Figure 3, removing SL negatively affects the game score. This is consistent with the earlier observations (Ammanabrolu and Hausknecht, 2020) reporting that KG-A2C without SL achieves no game score in ZORK1. However, as we can observe, TAC manages to retain some game score, which could be reflective of the positive role of ϵ-admissible exploration, inducing similar behaviour to SL.

From the observation that the absence of SL degrades the performance, we hypothesize that SL induces a regularization effect. We ran experiments with various strengths of supervised signals by increasing $\lambda_{\mathbb{T}}$ and $\lambda_{\mathbb{O}}$ in LUDICORP and TEMPLE, in which TAC attains higher scores at training compared with testing. As seen in Figure 4 (left two plots), higher $\lambda_{\mathbb{T}}$ and $\lambda_{\mathbb{O}}$ relaxes over-fitting, reaching the score from 7.7 to 15.8 in LUDICORP and from 5.8 to 8.0 in TEMPLE. Since SL is not directly related to rewards, this supports that SL acts as regularization. Further experimental results on ZORK1 is in Appendix H.

To further examine the role of admissible actions in SL, we hypothesize that SL is responsible for guiding the agent in the case that the reward signal is not collected. To verify this, we excluded ϵ-admissible exploration and ran TAC with different $\lambda_{\mathbb{T}}$ and $\lambda_{\mathbb{O}}$ in REVERB and ZORK1, in which TAC fails to achieve any score. According to Figure 4 (right two plots), TAC with stronger SL and $\epsilon = 0.0$ achieves game scores from 0 to 8.3 in REVERB, and from 0 to 18.3 in ZORK1, which suggests that SL acts as guidance. However, in the absence of ϵ-admissible exploration, despite the stronger supervised signals, TAC cannot match the scores using ϵ-admissible exploration.

Admissible Action Space During Training. To examine if constraining the action space to admissible actions during training leads to better utilization, we ran an ablation by masking template and object with admissible actions at training time. This leads to only generating admissible actions. Our plots in Figure 3 show that there is a reduction in the game score in PENTARI, REVERB and ZORK1 while DETECTIVE and ZORK3 observe slight to substantial increases, respectively. We speculate that the performance decay is due to the exposure bias (Bengio et al., 2015) introduced from fully constraining the action space to admissible actions during training. This means the agent does not learn how to act when it receives observations from inadmissible actions at test phase. However, for games like ZORK3, where the agent must navigate through the game to acquire sparse rewards, this technique seems to help.

Game	Kitchen. On the table is an elongated brown sack, smelling of hot peppers. A bottle is sitting on the table. The glass bottle contains: A quantity of water.
Inventory	You are carrying: A painting, A brass lantern (providing light)
Room	Kitchen. You are in the kitchen of the white house. A table seems to have been used recently for the preparation of food. A passage leads to the west and a dark staircase can be seen leading upward. A dark chimney leads down and to the east is a small window which is open. On the table is an elongated brown sack, smelling of hot peppers. A bottle is sitting on the table. The glass bottle contains: A quantity of water
LM Actions	'close bottle', 'close door', 'down', 'drink water', 'drop bottle', 'drop painting', 'east', 'empty bottle', 'get all', 'get bottle', 'get on table', 'get painting', 'get sack', 'north', 'open bottle', 'out', 'pour water on sack', 'put candle in sack', 'put painting in sack', 'put painting on sack', 'put water in sack', 'south', 'take all', 'take bottle', 'take painting', 'take sack', 'throw painting', 'up', 'wait', 'west'
KG Objects	'a', 'all', 'antique', 'board', 'bottle', 'brass', 'chimney', 'dark', 'door', 'down', 'east', 'exit', 'front', 'grue', 'house', 'is', 'kitchen', 'lantern', 'large', 'light', 'narrow', 'north', 'of', 'passage', 'path', 'quantity', 'rug', 'south', 'staircase', 'table', 'to', 'trap', 'trophy', 'up', 'west', 'white', 'window', 'with'
Admiss. Actions	'close window', 'east', 'jump', 'open bottle', 'open sack', 'put down all', 'put down light', 'put down painting', 'put light on table', 'put out light', 'put painting on table', 'take all', 'take bottle', 'take sack', 'throw light at window', 'up', 'west'

Table 2: Action space for a game observation (top panel) for CALM (LM), KG-A2C (KG), and the Admissible Action sets. Red and blue colored actions are the actions missed by either CALM or KG-A2C. Brown are the actions missed by both, and blacks are actions covered by both.

5.3 Qualitative Analysis

In this section, we show how CALM and KG-A2C restrict their action space. Table 2 shows a snippet of the gameplay in ZORK1. Top three rows are the textual observations and the bottom three rows are the actions generated by CALM, the objects extracted from KG in KG-A2C, and the admissible actions from the environment. CALM produces 30 different actions, but still misses 10 actions out of 17 admissible actions. Since DRRN learns to estimate Q value over generated 30 actions, those missing admissible actions can never be selected, resulting in a lack of exploration. On the other hand, KG-generated objects do not include 'sack' and 'painting', which means that the KG-A2C masks these two objects out from their object space. Then, the agent neglects any action that includes these two object, which also results

in a lack of exploration.

6 Discussion

Supervised Learning Loss. Intuitively, RL is to teach the agent *how to complete the game* while SL is to teach *how to play the game*. If the agent never acquired any reward signal, learning is only guided by SL. This is equivalent to applying imitation learning to the agent to follow more probable actions, a.k.a. admissible actions in TGs. However, in the case where the agent has reward signals to learn from, SL turns into regularization (§5.2), inducing a more uniformly distributed policies. In this sense, SL could be considered as the means to introduce the effects similar to entropy regularization in Ammanabrolu and Hausknecht (2020).

Exploration as Data Collection. In RL, the algorithm naturally collects and learns from data. Admissible action prediction from LM is yet to be accurate enough to replace the true admissible actions (Ammanabrolu and Riedl, 2021; Yao et al., 2020). This results in poor exploration and the agent may potentially never reach a particular state. On the other hand, KG-based methods (Ammanabrolu and Hausknecht, 2020; Peng et al., 2021; Xu et al., 2020, 2021, 2022; Ryu et al., 2022) must learn admissible actions before exploring the environment meaningfully. This will waste many samples since the agent will attempt inadmissible actions, collecting experiences of the unchanged states. Additionally, its action selection is largely dependent on the quality of KG. The missing objects from KG may provoke the same effects as LM, potentially obstructing navigating to a particular state. In this regards, ϵ-admissible exploration can overcome the issue by promoting behaviour that the agent would take after learning admissible actions fully. Under such conditions that a compact list of actions is either provided the environment or extracted by algorithm (Hausknecht et al., 2020), our approach can be employed. Intuitively, this is similar to playing the game with a game manual but not a ground truth to complete the game, which leads to collecting more meaningful data. It also collects more diverse data due to the stochasticity of exploration. Hence, TAC with ϵ-admissible exploration can learn *how to complete the game* with minimal knowledge of *how to play the game*.

Bias in Exploration. Our empirical results from adaptive ϵ experiments in Appendix I suggest that reasonable ϵ is required for both under-explored states and well-explored states. This could indicate that diverse data collection is necessary regardless of how much the agent knows about the game while ϵ value should not be too high such that the agent can exploit. Finally, from our ablation, fully constraining action space to admissible actions degrades performance. This could be a sign of exposure bias, which is a typical issue in NLG tasks (He et al., 2019; Mandya et al., 2020) and occurs between the training-testing discrepancy due to the teacher-forcing done at training (He et al., 2019). In our setting, this phenomena could potentially occur if the agent only learns from admissible actions at training time. Since ϵ-admissible exploration allows a collection of experiences of any actions (i.e., potentially inadmissible actions) with probability of $1 - \epsilon$, TAC with reasonable ϵ learns from high quality and unbiased data. Our observations indicate that both the algorithm that learns from data, and the exploration to acquire data are equally important.

7 Conclusion

Text-based Games (TGs) offer a unique framework for developing RL agents for goal-driven and contextually-aware natural language generation tasks. In this paper we took a fresh approach in utilizing the information from the TG environment, and in particular the admissibility of actions during the exploration phase of RL agent. We introduced a language-based actor critic method (TAC) with a simple ϵ-admissible exploration. The core of our algorithm is the utilization of admissible actions in training phase to guide the agent exploration towards collecting more informed experiences. Compared to state-of-the-art approaches with more complex design, our light TAC design achieves substantially higher game scores across 10-29 games.

We provided insights into the role of action admissibility and supervision signals during training and the implications at test phase for an RL agent. Our analysis showed that supervised signals towards admissible actions act as guideline in the absence of reward signal, while serving a regularization role in the presence of such signal. We demonstrated that reasonable ϵ probability threshold is required for high quality unbiased experience collection during the exploration phase.

References

Prithviraj Ammanabrolu and Matthew J. Hausknecht. 2020. Graph constrained reinforcement learning for natural language action spaces. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Prithviraj Ammanabrolu and Mark Riedl. 2021. Modeling worlds in text. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*.

Prithviraj Ammanabrolu, Ethan Tien, Zhaochen Luo, and Mark O. Riedl. 2020. How to avoid being eaten by a grue: Exploration strategies for text-adventure agents. *CoRR*, abs/2002.08795.

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 344–354, Beijing, China. Association for Computational Linguistics.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 1171–1179, Cambridge, MA, USA. MIT Press.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. COMET: commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4762–4779. Association for Computational Linguistics.

Marc-Alexandre Côté, Ákos Kádár, Xingdi (Eric) Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. 2018. Textworld: A learning environment for text-based games. In *Computer Games Workshop at ICML/IJCAI 2018*, pages 1–29.

Meng Fang, Yuan Li, and Trevor Cohn. 2017. Learning how to active learn: A deep reinforcement learning approach. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 595–605, Copenhagen, Denmark. Association for Computational Linguistics.

Scott Fujimoto, Herke van Hoof, and David Meger. 2018. Addressing function approximation error in actor-critic methods. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1587–1596. PMLR.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 1856–1865. PMLR.

Matthew J. Hausknecht, Prithviraj Ammanabrolu, Marc-Alexandre Côté, and Xingdi Yuan. 2020. Interactive fiction games: A colossal adventure. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7903–7910. AAAI Press.

Tianxing He, Jingzhao Zhang, Zhiming Zhou, and James R. Glass. 2019. Quantifying exposure bias for neural language generation. *CoRR*, abs/1905.10617.

Youngsoo Jang, Seokin Seo, Jongmin Lee, and Kee-Eung Kim. 2021. Monte-carlo planning and learning with language action value estimates. In *International Conference on Learning Representations*.

James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2016. Overcoming catastrophic forgetting in neural networks. *CoRR*, abs/1612.00796.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Angrosh Mandya, James O' Neill, Danushka Bollegala, and Frans Coenen. 2020. Do not let the history haunt you: Mitigating compounding errors in conversational question answering. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 2017–2025, Marseille, France. European Language Resources Association.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through

deep reinforcement learning. *Nat.*, 518(7540):529–533.

Xiangyu Peng, Mark O. Riedl, and Prithviraj Ammanabrolu. 2021. Inherently explainable reinforcement learning in natural language. *CoRR*, abs/2112.08907.

Dongwon Ryu, Ehsan Shareghi, Meng Fang, Yunqiu Xu, Shirui Pan, and Reza Haf. 2022. Fire burns, sword cuts: Commonsense inductive bias for exploration in text-based games. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 515–522, Dublin, Ireland. Association for Computational Linguistics.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2016. Prioritized experience replay. In *ICLR (Poster)*.

Xingyou Song, Yiding Jiang, Stephen Tu, Yilun Du, and Behnam Neyshabur. 2020. Observational overfitting in reinforcement learning. In *International Conference on Learning Representations*.

Jens Tuyls, Shunyu Yao, Sham M. Kakade, and Karthik R Narasimhan. 2022. Multi-stage episodic control for strategic exploration in text games. In *International Conference on Learning Representations*.

Yunqiu Xu, Meng Fang, Ling Chen, Yali Du, and Chengqi Zhang. 2021. Generalization in text-based games via hierarchical reinforcement learning. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1343–1353, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yunqiu Xu, Meng Fang, Ling Chen, Yali Du, Joey Zhou, and Chengqi Zhang. 2022. Perceiving the world: Question-guided reinforcement learning for text-based games. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 538–560, Dublin, Ireland. Association for Computational Linguistics.

Yunqiu Xu, Meng Fang, Ling Chen, Yali Du, Joey Tianyi Zhou, and Chengqi Zhang. 2020. Deep reinforcement learning with stacked hierarchical attention for text-based games. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Shunyu Yao, Rohan Rao, Matthew Hausknecht, and Karthik Narasimhan. 2020. 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)*, pages 8736–8754, Online. Association for Computational Linguistics.

Appendices

In this section, we provide the details of TAC, training, and full experimental results. We also provide Limitations and Ethical Considerations.

A Hyperparameters

Table 3 shows the hyper-parameters used for our experiments. For 905, ADVENT, ANCHOR, AWAKEN, DEEPHOME, INHUMANE and MOONLIT, gradients exploding has been observed with the hyper-parameters in Table 3, so we reduced learning rate to 10^{-5} for these games.

Training	
# of parallel environments	32
p_{va}	0.3
Optimization	
Batch size	64
Learning rate	10^{-4}
Weight decay	10^{-6}
Clip	5
γ	0.95
τ	0.001
Parameter size	
Word embedding dimension	100
Hidden dimension	128
Replay buffer	
Memory size	10^5
α	0.7
β	0.3
Weights for objectives	
λ_{R}	1.0
λ_{V}	1.0
λ_{Q}	1.0
$\lambda_{\mathbb{T}}$	1.0
$\lambda_{\mathbb{O}}$	1.0

Table 3: Hyper-parameters for main experiments.

B Parameter Size for ZORK1

The total parameter size of TAC in ZORK1 is 1,783,849 with 49,665 target state critic, which slightly varies by the size of template and object space per game. This is much lower than KG-A2C (4,812,741), but little higher than DRRN (1,486,081).[5]

C Training Time

We used Intel Xeon Gold 6150 2.70 GHz for CPU and Tesla V100-PCIE-16GB for GPU, 8 CPUs with 25GB memory, to train KG-A2C and TAC on ZORK1. The results are demonstrated in Table 5.[6] Our TAC has approximately three times lesser parameters than KG-A2C in ZORK1, which would be consistent across different games. On the other hand, for step per second, TAC is twice faster in GPU and thrice faster in CPU than KG-A2C. Approximated days for training TAC on CPU and GPU are 1.2 and 0.8 days while KG-A2C is 4.1 and 1.6 days. TAC still benefits from GPU, but not as much as KG-A2C as its training time is more dependent to the game engine than back-propagation.

[5]The code for KG-A2C is in https://github.com/rajammanabrolu/KG-A2C, and DRRN is in https://github.com/microsoft/tdqn.

[6]The code for KG-A2C is in https://github.com/rajammanabrolu/KG-A2C.

Table 4: Parameter size for ZORK1.

Name	Size
text_encoder_network.embedding.weight	[8000,100]
text_encoder_network.embedding_sa.weight	[4,128]
text_encoder_network.encoder.weight_ih_l0	[384,100]
text_encoder_network.encoder.weight_hh_l0	[384,128]
text_encoder_network.encoder.bias_ih_l0	[384]
text_encoder_network.encoder.bias_hh_l0	[384]
state_network.embedding_score.weight	[1024,128]
state_network.tf.weight	[128,384]
state_network.tf.bias	[128]
state_network.fc1.weight	[128,128]
state_network.fc1.bias	[128]
state_network.fc2.weight	[128,128]
state_network.fc2.bias	[128]
state_network.fc3.weight	[128,128]
state_network.fc3.bias	[128]
state_network.s.weight	[128,128]
state_network.s.bias	[128]
state_critic.fc1.weight	[128,128]
state_critic.fc1.bias	[128]
state_critic.fc2.weight	[128,128]
state_critic.fc2.bias	[128]
state_critic.fc3.weight	[128,128]
state_critic.fc3.bias	[128]
state_critic.v.weight	[1,128]
state_critic.v.bias	[1]
actor_network.fc1.weight	[128,128]
actor_network.fc1.bias	[128]
actor_network.fc2.weight	[128,128]
actor_network.fc2.bias	[128]
actor_network.fc3.weight	[128,128]
actor_network.fc3.bias	[128]
actor_network.a.weight	[128,128]
actor_network.a.bias	[128]
state_action_critic_1.fc1.weight	[128,256]
state_action_critic_1.fc1.bias	[128]
state_action_critic_1.fc2.weight	[128,128]
state_action_critic_1.fc2.bias	[128]
state_action_critic_1.fc3.weight	[128,128]
state_action_critic_1.fc3.bias	[128]
state_action_critic_1.q.weight	[1,128]
state_action_critic_1.q.bias	[1]
state_action_critic_2.fc1.weight	[128,256]
state_action_critic_2.fc1.bias	[128]
state_action_critic_2.fc2.weight	[128,128]
state_action_critic_2.fc2.bias	[128]
state_action_critic_2.fc3.weight	[128,128]
state_action_critic_2.fc3.bias	[128]
state_action_critic_2.q.weight	[1,128]
state_action_critic_2.q.bias	[1]
target_state_critic.fc1.weight	[128,128]
target_state_critic.fc1.bias	[128]
target_state_critic.fc2.weight	[128,128]
target_state_critic.fc2.bias	[128]
target_state_critic.fc3.weight	[128,128]
target_state_critic.fc3.bias	[128]
target_state_critic.v.weight	[1,128]
target_state_critic.v.bias	[1]
template_decoder_network.tmpl_gru.weight_ih_l0	[384,128]
template_decoder_network.tmpl_gru.weight_hh_l0	[384,128]
template_decoder_network.tmpl_gru.bias_ih_l0	[384]
template_decoder_network.tmpl_gru.bias_hh_l0	[384]
template_decoder_network.fc2.weight	[128,128]
template_decoder_network.fc2.bias	[128]
template_decoder_network.tmpl.weight	[235,128]
template_decoder_network.tmpl.bias	[235]
object_decoder_network.obj_gru.weight_ih_l0	[384,256]
object_decoder_network.obj_gru.weight_hh_l0	[384,128]
object_decoder_network.obj_gru.bias_ih_l0	[384]
object_decoder_network.obj_gru.bias_hh_l0	[384]
object_decoder_network.fc2.weight	[128,128]
object_decoder_network.fc2.bias	[128]
object_decoder_network.obj.weight	[699,128]
object_decoder_network.obj.bias	[699]

	Step/second (CPU)	Step/second (GPU)	Parameter Size
KG-A2C	0.28	0.71	4.8M
TAC	0.99	1.43	1.8M

Table 5: Training time as step per second in CPU and GPU and total parameter size for ZORK1.

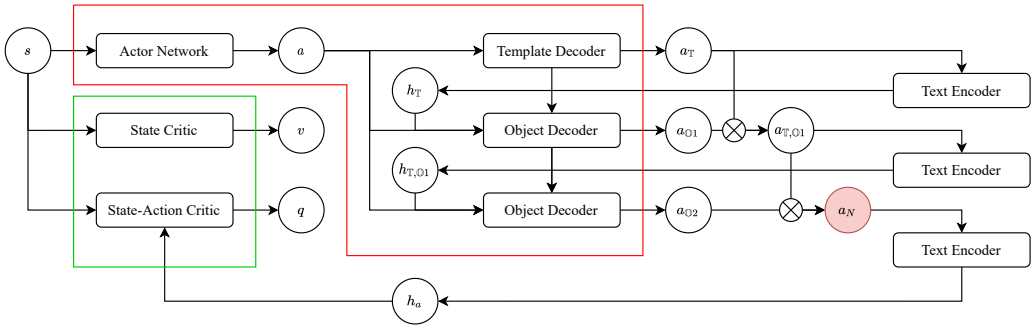

Figure 5: The details of actor and critic of text-based actor-critic; State representation is the input to actor-critic while a red circle is the output from actor, a_N representing natural language action. Red and green boxes indicate actor and critic, respectively.

D Details of Actor and Critic Components

Consider an action example (`take OBJ from OBJ`, `egg`, `fridge`) as (template, first object, second object). Template $a_{\mathbb{T}} =$ (`take OBJ from OBJ`) is sampled from template decoder and encoded to $h_{\mathbb{T}}$ with text encoder. Object decoder takes action representation a and encoded semi-completed action $h_{\mathbb{T}}$ and produces the first object $a_{\mathbb{O}1} =$ (`egg`). The template $a_{\mathbb{T}} =$ (`take OBJ from OBJ`) and the first object $a_{\mathbb{O}1} =$ (`egg`) are combined to $a_{\mathbb{T},\mathbb{O}1} =$ (`take egg from OBJ`), $a_{\mathbb{T}} \otimes a_{\mathbb{O}1} = a_{\mathbb{T},\mathbb{O}1}$. $a_{\mathbb{T},\mathbb{O}1}$ is then, encoded to hidden state $h_{\mathbb{T},\mathbb{O}1}$ with text encoder. Similarly, the object decoder takes a and $h_{\mathbb{T},\mathbb{O}1}$ and produces the second object $a_{\mathbb{O}2} =$ (`fridge`). $a_{\mathbb{T},\mathbb{O}1}$ and $a_{\mathbb{O}2}$ are combined to be natural language action, $a_{\mathbb{T},\mathbb{O}1} \otimes a_{\mathbb{O}2} = a_N$ Finally, a_N is encoded to h_a with text encoder and inputted to state-action critic to predict Q value.

E Comparison with Vanilla A2C in Ammanabrolu and Hausknecht (2020)

Architecture. Vanilla A2C from Ammanabrolu and Hausknecht (2020) uses separate gated recurrent units (GRUs) to encode textual observations and previous action, $(o_{\text{game}}, o_{\text{look}}, o_{\text{inv}}, a_{t-1})$, and transforms the game score, n_{score}, into binary encoding. Then, they are concatenated and passed through state network to form state representation. Their state network is GRU-based to account historical information. The actor-critic network consists of actor and state value critic, so the state representation is used to estimate state value and produce the policy distribution.

Our TAC uses a single shared GRU to encode textual observations and previous action with different initial state to signify that the text encoder constructs the general representation of text while the game score is embedded to learnable high dimentional vector. However, when constructing state representation, we only used $(o_{\text{game}}, o_{\text{look}}, o_{\text{inv}})$ under our observation that o_{game} carries semantic information about a_{t-1}. Additionally, we also observed that the learned game score representation acts as conditional vector in Appendix F, so the state representation is constructed as an instance of observation without historical information. Finally, we included additional modules, state-action value critic (Haarnoja et al., 2018), target state critic (Mnih et al., 2015) and two state-action critics (Fujimoto et al., 2018; Haarnoja et al., 2018) for practical purpose.

Objective Function. Three objectives are employed in Ammanabrolu and Hausknecht (2020), reinforcement learning (RL), supervised learning (SL) and entropy regularization. Both RL and SL are also used in our objectives with minor changes in value function update in RL. That is, two state-action value critics are updated independently to predict Q value per state-action pair and target state critic is updated as moving average of state critic Notable difference is that we excluded entropy regularization from Ammanabrolu and Hausknecht (2020). This is because under our ablation in Section 5.2, we observed that SL acts as regularization.

Replay Buffer Unlike on-policy vanilla A2C (Ammanabrolu and Hausknecht, 2020), since TAC utilizes ϵ-admissible exploration, it naturally sits as off-policy algorithm. We used prioritized experience replay (PER) as our replay buffer (Schaul et al., 2016). Standard PER assigns a newly acquired experience with the maximum priority. This enforces the agent to prioritize not-yet-sampled ex-

Case 1.1
Step: 4 Game: Kitchen You are in the kitchen of the white house. A table seems to have been used recently for the preparation of food. A passage leads to the west and a dark staircase can be seen leading upward. A dark chimney leads down and to the east is a small window which is open. On the table is an elongated brown sack, smelling of hot peppers. A bottle is sitting on the table. The glass bottle contains: A quantity of water Look: Kitchen You are in the kitchen of the white house. A table seems to have been used recently for the preparation of food. A passage leads to the west and a dark staircase can be seen leading upward. A dark chimney leads down and to the east is a small window which is open. On the table is an elongated brown sack, smelling of hot peppers. A bottle is sitting on the table. The glass bottle contains: A quantity of water Inv: You are empty handed. Score: 10 Action: west
Case 1.2
Step: 15 Game: Kitchen On the table is an elongated brown sack, smelling of hot peppers. A bottle is sitting on the table. The glass bottle contains: A quantity of water Look: Kitchen You are in the kitchen of the white house. A table seems to have been used recently for the preparation of food. A passage leads to the west and a dark staircase can be seen leading upward. A dark chimney leads down and to the east is a small window which is open. On the table is an elongated brown sack, smelling of hot peppers. A bottle is sitting on the table. The glass bottle contains: A quantity of water Inv: You are carrying: A painting A brass lantern (providing light) Score: 39 Action: west
Case 1.3
Step: 20 Game: Kitchen On the table is an elongated brown sack, smelling of hot peppers. A bottle is sitting on the table. The glass bottle contains: A quantity of water Look: Kitchen You are in the kitchen of the white house. A table seems to have been used recently for the preparation of food. A passage leads to the west and a dark staircase can be seen leading upward. A dark chimney leads down and to the east is a small window which is open. On the table is an elongated brown sack, smelling of hot peppers. A bottle is sitting on the table. The glass bottle contains: A quantity of water Inv: You are empty handed. Score: 45 Action: east

Table 6: Case 1; Game observation and the selected action snippets from ZORK1.

Case 1.1						
	$n_{\text{score}} = 10$		$n_{\text{score}} = 39$		$n_{\text{score}} = 45$	
	$\pi(a_{\mathbb{T}}\|o)$	$Q(o,a)$	$\pi(a_{\mathbb{T}}\|o)$	$Q(o,a)$	$\pi(a_{\mathbb{T}}\|o)$	$Q(o,a)$
west	**0.9998**	**23.7460**	0.000	4.1434	0.000	5.0134
east	0.000	18.4385	**0.5674**	**5.1640**	**0.9996**	**6.0319**
Case 1.2						
	$n_{\text{score}} = 10$		$n_{\text{score}} = 39$		$n_{\text{score}} = 45$	
	$\pi(a_{\mathbb{T}}\|o)$	$Q(o,a)$	$\pi(a_{\mathbb{T}}\|o)$	$Q(o,a)$	$\pi(a_{\mathbb{T}}\|o)$	$Q(o,a)$
west	**0.9975**	**27.6005**	**0.9819**	**8.3794**	**0.8967**	**8.0586**
east	0.000	23.6015	0.0002	6.5284	0.000	6.4848
Case 1.3						
	$n_{\text{score}} = 10$		$n_{\text{score}} = 39$		$n_{\text{score}} = 45$	
	$\pi(a_{\mathbb{T}}\|o)$	$Q(o,a)$	$\pi(a_{\mathbb{T}}\|o)$	$Q(o,a)$	$\pi(a_{\mathbb{T}}\|o)$	$Q(o,a)$
west	**0.7872**	**22.2419**	0.0001	4.9664	0.000	5.0169
east	0.0055	19.1751	**0.7821**	**5.7299**	**0.9999**	**6.2653**

Table 7: Case 1; The changes in policy and Q value based on the score embedding from ZORK1.

periences over others. As we are using 32 parallel environments and 64 batch size for update, half of the updates will be directed by newly acquired experiences, which not all of them may be useful. Thus, instead, we assign newly acquired experience with TD errors when they are added to the buffer. This risks the agent not using some experiences, but it is more efficient since we sample useful batch of experiences.

F Qualitative Analysis

It has been repetitively reported that including game score when constructing state helps in TGs (Ammanabrolu and Hausknecht, 2020; Jang et al., 2021). Here, we provide some insights in what the agent learns from the observations using fully trained TAC. To illustrate this, we highlight the role of game score on the action preference of the TAC for the same observation in ZORK1. Observations for different cases can be found in Table 6 and Table 8 while the policy and Q value are in Table 7 and Table 9.

Case 1 in Table 6 and Table 7 For three different cases, `Case 1.1`, `Case 1.2`, and `Case 1.3`, the agent is at `Kitchen` location, so many semantic meaning between textual observations are similar, i.e. o_{look} or o_{inv}. For each case, the agent is meant to go `west` with $n_{\text{score}} = 10$, go `west` with $n_{\text{score}} = 39$, and go `east` with $n_{\text{score}} = 45$, respectively. In `Case 1.1`, despite the optimal choice of action is `west`, by replacing the score from $n_{\text{score}} = 10$ to $n_{\text{score}} = 45$, the agent chooses `east`, which is appropriate for `Case 1.3`. Another interesting observation is that replacing game score decreases Q value from 23.7460 to 5.0134 for `west` and from 18.4385 to 6.0319 for `east` in `Case 1.1`. This seems like the agent thinks it has already acquired reward signals between $n_{\text{score}} = 10$ and $n_{\text{score}} = 45$, resulting in a reduction in Q value. We speculate that this is because the embedding

Case 2.1
Step: 2 Game: Behind House You are behind the white house. A path leads into the forest to the east. In one corner of the house there is a small window which is slightly ajar. Look: Behind House You are behind the white house. A path leads into the forest to the east. In one corner of the house there is a small window which is slightly ajar. Inv: You are empty handed. Score: 0 Action: open window
Case 2.2
Step: 3 Game: With great effort, you open the window far enough to allow entry. Look: Behind House You are behind the white house. A path leads into the forest to the east. In one corner of the house there is a small window which is open. Inv: You are empty handed. Score: 0 Action: west
Case 2.3
Step: 21 Game: Behind House Look: Behind House You are behind the white house. A path leads into the forest to the east. In one corner of the house there is a small window which is open. Inv: You are empty handed. Score: 45 Action: north

Table 8: Case 2; Game observation and the selected action snippets from ZORK1.

Case 2.1				
	$n_{\text{score}} = 0$		$n_{\text{score}} = 45$	
	$\pi(a_{\mathbb{T}}\|o)$	$Q(o,a)$	$\pi(a_{\mathbb{T}}\|o)$	$Q(o,a)$
open window	**0.9999**	**29.0205**	0.0111	5.9599
west	0.0000	28.6848	0.0893	6.1119
north	0.0000	26.7997	**0.8174**	**6.2819**
Case 2.2				
	$n_{\text{score}} = 0$		$n_{\text{score}} = 45$	
	$\pi(a_{\mathbb{T}}\|o)$	$Q(o,a)$	$\pi(a_{\mathbb{T}}\|o)$	$Q(o,a)$
open window	0.0000	30.2154	0.0000	6.1354
west	**0.9999**	**32.0298**	0.0000	5.8312
north	0.0000	26.7509	**0.9952**	**6.6669**
Case 2.3				
	$n_{\text{score}} = 0$		$n_{\text{score}} = 45$	
	$\pi(a_{\mathbb{T}}\|o)$	$Q(o,a)$	$\pi(a_{\mathbb{T}}\|o)$	$Q(o,a)$
open window	0.0000	30.2184	0.0001	6.0443
west	**0.9999**	**32.0302**	0.0000	5.6724
north	0.0000	26.7494	**0.9867**	**6.5545**

Table 9: Case 2; The changes in policy and Q value based on the score embedding from ZORK1.

of n_{score} carries some inductive bias, i.e. temporal, for the agent to infer the stage of the game. This is consistently manifested in `Case 1.3`, but in `Case 1.2`, the agent is robust to the game score because it carries `painting` that is directly related to reward signals, navigating to pursue that particular reward, which is `put paining in case` for reward signal of $+6$ in `Living Room` location.

Case 2 in Table 8 and Table 9 In `Case 2`, the agent is at `Behind House` for three other sets of game instances, which has action and score pair as, `open window` for $n_{\text{score}} = 0$, `west` for $n_{\text{score}} = 0$, and `north` for $n_{\text{score}} = 45$. The phenomenon between `Case 1.1` and `Case 1.3` occurs the same for `Case 2.2` and `Case 2.3`. However, unlike `Case 1`, the score between `Case 2.1` and `Case 2.2` is the same. This means that the agent somehow chooses the optimal action for `Case 2.2` over `Case 2.1` in the case where $n_{\text{score}} = 0$ is injected for `Case 2.3`. This appears to be that the agent can capture semantic correlation between `"In one corner of the house there is a small window which is open"` from textual observation in `Case 2.3` and `open window` action. Because `a small window` is already opened, `open window` action is no longer required, so the agent tends to produce `west`, which is appropriate for `Case 2.2`.

Thus, from our qualitative analysis, we speculate that the agent captures the semantics of the textual observations and infers the game stage from game score embedding to make optimal decision.

G Full Experimental Results

The full learning curve of TAC and game score comparison are presented in Figure 6 and Table 10.

H Stronger Supervised Signals for ZORK1

We also explored how stronger supervised signals can induce better regularization in ZORK1. Similar to other sets of experiments, we selected variety of

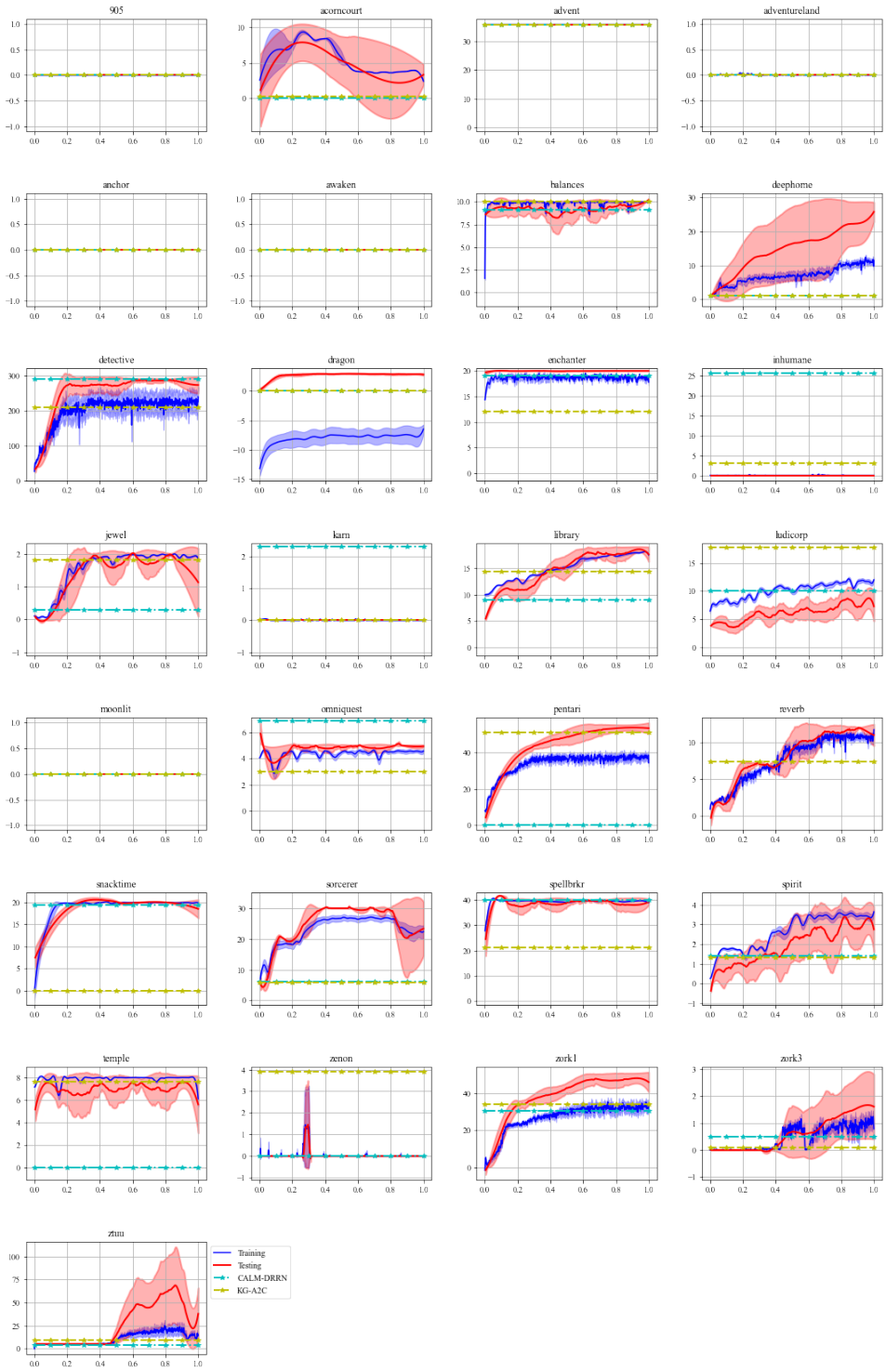

Figure 6: The full learning curve for TAC, compared with TDQN and KG-A2C

	NAIL	DRRN	TDQN	CALM-DRRN	KG-A2C	TAC
905	0.0	0.0	0.0	0.0	0.0	0.0 ± 0.0
ACORNCOURT	0.0	*10.0*	1.6	0.0	0.3	**3.4 ± 1.6**
ADVENT †	36.0	36.0	36.0	36.0	36.0	36.0 ± 0.0
ADVENTURELAND	0.0	*20.6*	0.0	0.0	0.0	0.0 ± 0.0
ANCHOR	0.0	0.0	0.0	0.0	0.0	0.0 ± 0.0
AWAKEN	0.0	0.0	0.0	0.0	0.0	0.0 ± 0.0
BALANCES	*10.0*	*10.0*	4.8	9.1	***10.0***	***10.0 ± 0.1***
DEEPHOME ‡	13.3	1.0	1.0	1.0	1.0	***25.4 ± 3.2***
DETECTIVE	136.9	197.8	169.0	***289.7***	207.9	272.3 ± 23.3
DRAGON	0.6	-3.5	-5.3	0.1	0.0	***2.81 ± 0.15***
ENCHANTER	0.0	*20.0*	8.6	19.1	12.1	***20.0 ± 0.0***
INHUMANE	0.6	0.0	0.7	***25.7***	3.0	0.0 ± 0.0
JEWEL	1.6	1.6	0.0	0.3	***1.8***	1.17 ± 1.0
KARN	1.2	2.1	0.7	***2.3***	0.0	0.0 ± 0.0
LIBRARY	0.9	17.0	6.3	9.0	14.3	***18.0 ± 1.2***
LUDICORP	8.4	13.8	6.0	10.1	***17.8***	7.7 ± 2.5
MOONLIT	0.0	0.0	0.0	0.0	0.0	0.0 ± 0.0
OMNIQUEST	5.6	5.0	*16.8*	**6.9**	3.0	4.9 ± 0.1
PENTARI	0.0	27.2	17.4	0.0	50.7	***53.2 ± 2.9***
REVERB	0.0	8.2	0.3	–	7.4	***11 ± 1.4***
SNACKTIME	0.0	0.0	9.7	***19.4***	0.0	18.6 ± 2.0
SORCERER	5.0	20.8	5.0	6.2	5.8	***23.2 ± 9.3***
SPELLBRKR	*40.0*	37.8	18.7	***40.0***	21.3	39.0 ± 1.4
SPIRIT	1.0	0.8	0.6	1.4	1.3	***2.91 ± 1.1***
TEMPLE	7.3	7.4	*7.9*	0.0	**7.6**	5.8 ± 2.3
ZENON	0.0	0.0	0.0	0.0	***3.9***	0.0 ± 0.0
ZORK1	10.3	32.6	9.9	30.4	34	***46.3 ± 5.0***
ZORK3	*1.8*	0.5	0.0	0.5	0.1	**1.6 ± 1.2**
ZTUU	0.0	21.6	4.9	3.7	9.2	***33.2 ± 26.0***
MEAN	0.0536	0.1156	0.0665	0.0936	0.1094	***0.1560***

Table 10: Game score comparison over 29 game environments in Jericho, with best results highlighted by **boldface**. NAIL and DRRN are non-generative baselines while TDQN and KG-A2C are generative baselines. The last row is the mean game score over all the environments. The initial game score of ADVENT † is 36 and DEEPHOME ‡ is 1.

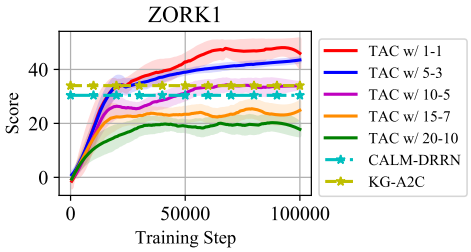

Figure 7: The learning curve of TAC for *regularization* ablation in ZORK1. Stronger supervised signals are used with $\epsilon = 0.3$, where 5-3 signifies $\gamma_{\mathbb{T}} = 5$ and $\gamma_{\mathbb{O}} = 3$.

$\lambda_{\mathbb{T}}$-$\lambda_{\mathbb{O}}$ pair. However, our results show that TAC starts under-fitting in ZORK1 when larger $\lambda_{\mathbb{T}}$ and $\lambda_{\mathbb{O}}$ are applied.

I Adaptive Score-based ϵ

We also designed the epsilon scheduler that dynamically assigns ϵ based on the game score that the agent has achieved; $\epsilon \propto e^{\frac{a_\epsilon}{n_{\text{tst}}} n_{\text{score}}}$, where a_ϵ is the hyper-parameters and n_{tst} is the average testing game score. During training, higher n_{score} exponentially increases ϵ while a_ϵ controls the slope of the exponential function. Higher a_ϵ makes the slope more steep. Intuitively, as the agent exploits the well-known states, ϵ is small, encouraging the agent to follow its own policy, and as the agent reaches the under-explored states (i.e., similar to test condition), ϵ increases to encourage more diversely. The ϵ is normalized and scaled. The example plot is shown in FIgure 10.

We conducted a set of ablations with dynamic ϵ value in DETECTIVE, PENTARI, REVERB, ZORK1 and ZORK3. We used $\epsilon_{\text{min}} = \{0.0, 0.3\}$, $a_\epsilon = \{3, 9\}$ and $\epsilon_{\text{max}} = \{0.7, 1.0\}$, so total 8 different hyper-parameters. Figure 8 shows fixed $\epsilon_{\text{min}} = 0.0$ with varying a_ϵ and ϵ_{max} and Figure 8 shows fixed $\epsilon_{\text{min}} = 0.3$. Other than ZORK3, TAC with dynamic ϵ matches or underperforms TAC with fixed $\epsilon = 0.3$. There are two interesting phenomenons. (i) Too high ϵ_{max} results in more unstable learning and lower performance. This becomes very obvious in PENTARI, REVERB and ZORK1, where regardless of ϵ_{min} and a_ϵ, if $\epsilon_{\text{max}} = 1.0$, the learning curve is relatively low. In DETECTIVE of Figure 8, the learning becomes much more unstable

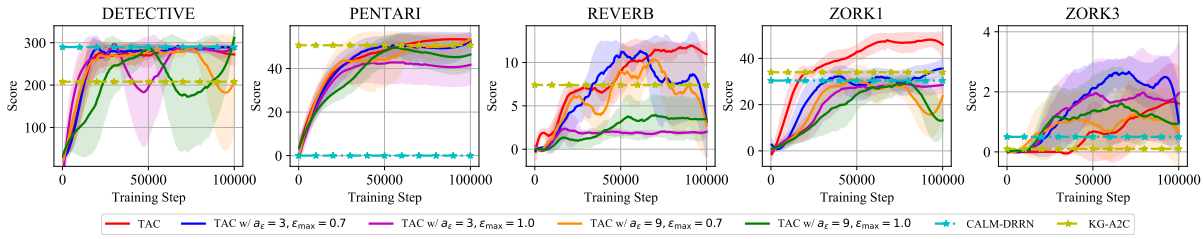

Figure 8: The learning curve of TAC with dynamic epsilon on five popular games. All the experiments were done with fixed $\epsilon_{\text{min}} = 0.0$, $a_\epsilon = \{3, 9\}$ and $\epsilon_{\text{max}} = \{0.7, 1.0\}$.

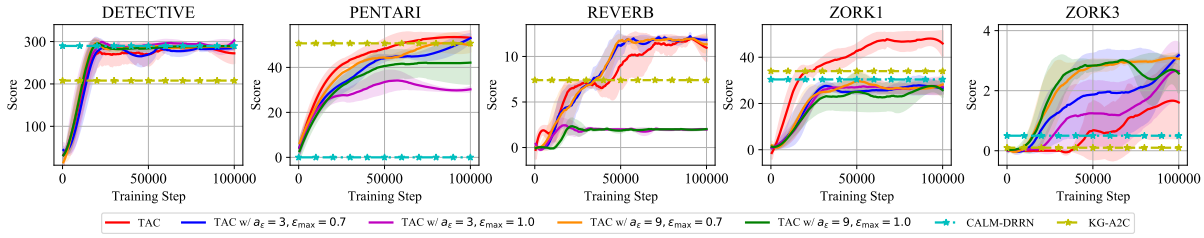

Figure 9: The learning curve of TAC with dynamic epsilon on five popular games. All the experiments were done with fixed $\epsilon_{\text{min}} = 0.3$, $a_\epsilon = \{3, 9\}$ and $\epsilon_{\text{max}} = \{0.7, 1.0\}$.

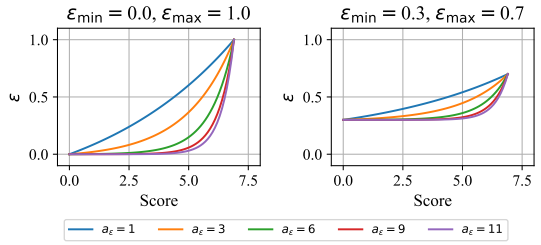

Figure 10: The exponential probability of ϵ over the game score. Left is with $\epsilon_{\text{min}} = 0.0, \epsilon_{\text{max}} = 1.0$ and right is with $\epsilon_{\text{min}} = 0.3, \epsilon_{\text{max}} = 0.7$ between the game score of 0 to 6. Five different a_ϵ is drawn per plot.

with $\epsilon_{\text{max}} = 1.0$. This indicates that even under-explored states, exploitation may still be required. (ii) Too low ϵ_{min} results in more unstable learning and lower performance. Although PENTARI benefits from $\epsilon_{\text{min}} = 0.0$, the learning curves in Figure 8 is generally lower and unstable than Figure 9. This appears to be that despite how much the agent learned the environment, it still needs to act stochastically to collect diverse experiences.

J Limitations

Similar to CALM-DRRN (Yao et al., 2020), KG-A2C (Ammanabrolu and Hausknecht, 2020) and KG-A2C variants (Ammanabrolu et al., 2020; Xu et al., 2020; Peng et al., 2021) that use admissible actions, our method still utilizes admissible actions. This makes our TAC not suitable for environments that do not provide admissible action set. In the absence of admissible actions, our TAC requires some prior knowledge of a compact set of more probable actions from LMs or other sources. This applies to other problems, for instance, applying our proposed method to language-grounded robots requires action candidates appropriate per state that they must be able to sample during training. The algorithm proposed by Hausknecht et al. (2020) extracts admissible actions by simulating thousands of actions per every step in TGs. This can be used to extract a compact set of actions in other problems, but it would not be feasible to apply if running a simulation is computationally expensive or risky (incorrect action in real-world robot may result in catastrophic outcomes, such as breakdown).

K Ethical Considerations

Our proposal may impact other language-based autonomous agents, such as dialogue systems or language-grounded robots. In a broader aspect, it contributes to the automated decision making, which can be used in corporation and government. When designing such system, it is important to bring morals and remove bias to be used as intended.