

Keep CALM and Explore: Language Models for Action Generation in Text-based Games

Shunyu Yao[†], Rohan Rao[†], Matthew Hausknecht[‡], Karthik Narasimhan[†]

[†]Princeton University [‡]Microsoft Research

{shunyuy, rohanr, karthikn}@princeton.edu

matthew.hausknecht@microsoft.com

Abstract

Text-based games present a unique challenge for autonomous agents to operate in natural language and handle enormous action spaces. In this paper, we propose the Contextual Action Language Model (CALM) to generate a compact set of action candidates at each game state. Our key insight is to train language models on human gameplay, where people demonstrate linguistic priors and a general *game sense* for promising actions conditioned on game history. We combine CALM with a reinforcement learning agent which re-ranks the generated action candidates to maximize in-game rewards. We evaluate our approach using the Jericho benchmark (Hausknecht et al., 2019a), on games *unseen* by CALM during training. Our method obtains a 69% relative improvement in average game score over the previous state-of-the-art model. Surprisingly, on half of these games, CALM is competitive with or better than other models that have access to ground truth admissible actions.*

1 Introduction

Text-based games have proven to be useful testbeds for developing agents that operate in language. As interactions in these games (input observations, action commands) are through text, they require solid language understanding for successful gameplay. While several reinforcement learning (RL) models have been proposed recently (Narasimhan et al., 2015; He et al., 2015; Hausknecht et al., 2019a; Ammanabrolu and Riedl, 2019), combinatorially large action spaces continue to make these games challenging for these approaches.

The action space problem is exacerbated by the fact that only a tiny fraction of action commands are admissible in any given game state. An admissible action is one that is parseable by the game

*Code and data are available at <https://github.com/princeton-nlp/calm-textgame>.

Observation: You are in the living room. There is a doorway to the east, a wooden door with strange gothic lettering to the west, which appears to be nailed shut, a trophy case, and a large oriental rug in the center of the room. You are carrying: A brass lantern ...

Random Actions:

close door, north a, eat troll with egg, ...

CALM (n-gram) Actions:

enter room, leave room, lock room, open door, close door, knock on door, ...

CALM (GPT-2) Actions:

east, open case, get rug, turn on lantern, move rug, unlock case with key, ...

Next Observation: With a great effort, the rug is moved to one side of the room, revealing the dusty cover of a closed trap door...

Figure 1: Sample gameplay from ZORK1 along with action sets generated by two variants of CALM. The game recognizes a vocabulary size of 697, resulting in more than $697^4 \approx 200$ billion potential 4-word actions. ‘*move rug*’ is the optimal action to take here and is generated by our method as a candidate.

engine and changes the underlying game state. For example, in Figure 1, one can observe that randomly sampling actions from the game vocabulary leads to several inadmissible ones like ‘*north a*’ or ‘*eat troll with egg*’. Thus, narrowing down the action space to admissible actions requires both syntactic and semantic knowledge, making it challenging for current systems.

Further, even within the space of admissible actions, it is imperative for an autonomous agent to know which actions are most promising to advance the game forward, and explore them first. Human players innately display such game-related common sense. For instance in Figure 1, players

might prefer the command “move rug” over “knock on door” since the door is nailed shut. However, even the state-of-the-art game-playing agents do not incorporate such priors, and instead rely on rule-based heuristics (Hausknecht et al., 2019a) or handicaps provided by the learning environment (Hausknecht et al., 2019a; Ammanabrolu and Hausknecht, 2020) to circumvent these issues.

In this work, we propose the Contextual Action Language Model (CALM) to alleviate this challenge. Specifically, at each game step we use CALM to generate action candidates, which are fed into a Deep Reinforcement Relevance Network (DRRN) (He et al., 2015) that uses game rewards to learn a value function over these actions. This allows our model to combine generic linguistic priors for action generation with the ability to adaptively choose actions that are best suited for the game.

To train CALM, we introduce a novel dataset of 426 human gameplay transcripts for 590 different text-based games. While these transcripts are noisy and actions are not always optimal, they contain a substantial amount of linguistic priors and game sense. Using this dataset, we train a single instance of CALM and deploy it to generate actions across many different downstream games. Importantly, in order to demonstrate the generalization of our approach, we do not use any transcripts from our evaluation games to train the language model.

We investigate both n-gram and state-of-the-art GPT-2 (Radford et al., 2019) language models and first evaluate the quality of generated actions in isolation by comparing against ground-truth sets of admissible actions. Subsequently, we evaluate the quality of CALM in conjunction with RL over 28 games from the Jericho benchmark (Hausknecht et al., 2019a). Our method outperforms the previous state-of-the-art method by 69% in terms of average normalized score. Surprisingly, on 8 games our method even outperforms competing methods that use the admissible action handicap – for example, in the game of INHUMANE, we achieve a score of 25.7 while the state-of-the-art KG-A2C agent (Ammanabrolu and Hausknecht, 2020) achieved 3.

In summary, our contributions are two-fold. First, we propose a novel learning-based approach for reducing enormous action spaces in text-based games using linguistic knowledge. Second, we introduce a new dataset of human gameplay transcripts, along with an evaluation scheme to measure the quality of action generation in these games.

2 Related Work

Reinforcement Learning for Text-based Games

Early work on text-based games (Narasimhan et al., 2015; He et al., 2015) developed RL agents on synthetic environments with small, pre-defined text action spaces. Even with small actions spaces (e.g. < 200 actions), approaches to filter inadmissible actions (Zahavy et al., 2018; Jain et al., 2019) led to faster learning convergence. Recently, Hausknecht et al. (2019a) introduced Jericho – a benchmark of challenging man-made text games. These games contain significantly greater linguistic variation and larger action spaces compared to frameworks like TextWorld (Côté et al., 2018).

To assist RL agents, Jericho provides a handicap that identifies admissible actions at each game state. This has been used by approaches like DRRN (He et al., 2015) as a reduced action space. Other RL agents like TDQN (Hausknecht et al., 2019a) and KGA2C (Ammanabrolu and Hausknecht, 2020) rely on the handicap for an auxiliary training loss. In general, as these RL approaches lack linguistic priors and only learn through in-game rewards, they are reliant on the admissible-action handicap to make the action space tractable to explore.

Linguistic Priors for Text-based Games

A different line of work has explored various linguistic priors for generating action commands. Fulda et al. (2017) used Word2vec (Mikolov et al., 2013) embeddings to infer affordance properties (i.e. verbs suitable for an object). Other approaches (Kostka et al., 2017; Hausknecht et al., 2019b) trained simple n-gram language models to learn affordances for action generation. Perhaps most similar to our work is that of Tao et al. (2018), who trained seq2seq (Sutskever et al., 2014) models to produce admissible actions in synthetic TextWorld (Côté et al., 2018) games. In a slightly different setting, Urbanek et al. (2019) trained BERT (Devlin et al., 2018) to generate contextually relevant dialogue utterances and actions in fantasy settings. However, these approaches are game-specific and do not use any reinforcement learning to optimize gameplay. In contrast, we combine strong linguistic priors with reinforcement learning, and use a modern language model that can generate complex actions and flexibly model the dependency between actions and contexts. We also train on multiple games and generalize to unseen games.

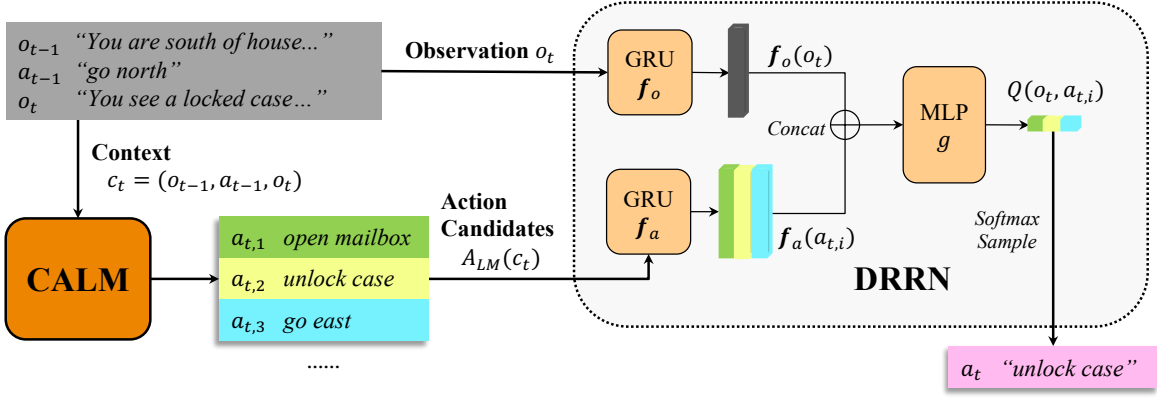


Figure 2: CALM combined with an RL agent – DRRN (He et al., 2015) – for gameplay. CALM is trained on transcripts of human gameplay for action generation. At each state, CALM generates action candidates conditioned on the game context, and the DRRN calculates the Q-values over them to select an action. Once trained, a single instance of CALM can be used to generate actions for any text-based game.

Generation in Text-based Games and Interactive Dialog Besides solving games, researchers have also used language models to *create* text-based games. Ammanabrolu et al. (2019) used Markov chains and neural language models to procedurally generate quests for TextWorld-like games. AI Dungeon 2 (Walton, 2019) used GPT-2 to generate narrative text in response to arbitrary text actions, but lacked temporal consistency over many steps.

More broadly, the concept of generating candidates and re-ranking has been studied in other interactive language tasks such as dialogue (Zhao and Eskenazi, 2016; Williams et al., 2017; Song et al., 2016; Chen et al., 2017) and communication games (Lazaridou et al., 2020). These approaches often focus on improving aspects like fluency and accuracy of the generated utterances, whereas our re-ranking approach only aims to maximize future rewards in the task. Also, our CALM pre-trained model generalizes to new environments without requiring any re-training.

3 Method

3.1 Background

A text-based game can be formally specified as a partially observable Markov decision process (POMDP) (S, T, A, O, R, γ) , where a player issues text actions $a \in A$ and receives text observations $o \in O$ and scalar rewards $r = R(s, a)$ at each step. Different games have different reward designs, but typically provide sparse positive rewards for solving key puzzles and advancing the story,

and negative rewards for dying. $\gamma \in [0, 1]$ is the reward discount factor. Latent state $s \in S$ contains the current game information (e.g. locations of the player and items, the player’s inventory), which is only partially reflected in o . The transition function $s' = T(s, a)$ specifies how action a is applied on state s , and a is **admissible** at state s if $T(s, a) \neq s$ (i.e. if it is parseable by the game and changes the state). S, T and R are not provided to the player.

Reinforcement Learning One approach to developing text-based game agents is reinforcement learning (RL). The Deep Reinforcement Relevance Network (DRRN) (He et al., 2015) is an RL algorithm that learns a Q-network $Q_\phi(o, a)$ parametrized by ϕ . The model encodes the observation o and each action candidate a using two separate encoders f_o and f_a (usually recurrent neural networks such as GRU (Cho et al., 2014)), and then aggregates the representations to derive the Q-value through a decoder g :

$$Q_\phi(o, a) = g(f_o(o), f_a(a)) \quad (1)$$

For learning ϕ , tuples (o, a, r, o') of observation, action, reward and the next observation are sampled from an experience replay buffer and the following temporal difference (TD) loss is minimized:

$$\mathcal{L}_{TD}(\phi) = (r + \gamma \max_{a' \in A} Q_\phi(o', a') - Q_\phi(o, a))^2 \quad (2)$$

During gameplay, a softmax exploration policy is used to sample an action:

$$\pi_\phi(a|o) = \frac{\exp(Q_\phi(o, a))}{\sum_{a' \in A} \exp(Q_\phi(o, a'))} \quad (3)$$

While the above equation contains only a single observation, this can also be extended to a policy $\pi(a|c)$ conditioned on a longer context $c = (o_1, a_1, \dots, o_t)$ of previous observations and actions till current time step t . Note that when the action space A is large, (2) and (3) become intractable.

3.2 Contextual Action Language Model (CALM)

To reduce large action spaces and make learning tractable, we train language models to generate compact sets of actions candidates. Consider a dataset \mathcal{D} of N trajectories of human gameplay across different games, where each trajectory of length l consists of interleaved observations and actions $(o_1, a_1, o_2, a_2, \dots, o_l, a_l)$. The *context* c_t at timestep t is defined as the history of observations and actions, i.e. $c_t = (o_1, a_1, \dots, a_{t-1}, o_t)$. In practice, we find that a window size of 2 works well, i.e. $c_t = (o_{t-1}, a_{t-1}, o_t)$. We train parametrized language models p_θ to generate actions a conditioned on contexts c . Specifically, we use all N trajectories and minimize the following cross-entropy loss:

$$\mathcal{L}_{\text{LM}}(\theta) = -\mathbb{E}_{(a,c) \sim \mathcal{D}} \log p_\theta(a|c) \quad (4)$$

Since each action a is typically a multi-word phrase consisting of m tokens a^1, a^2, \dots, a^m , we can further factorize the right hand side of (4) as:

$$p_\theta(a|c) = \prod_{i=1}^m p_\theta(a^i | a^{<i}, c) \quad (5)$$

Thus, we can simply use the cross-entropy loss over each token a^i in action a during training. We investigate two types of language models:

1. n-gram: This model simply uses n-gram counts from actions in \mathcal{D} to model the following probability:

$$p_{(n,\alpha)}(a^i | a^{<i}) = \frac{\text{cnt}(a^{i-n+1}, \dots, a^i) + \alpha}{\text{cnt}(a^{i-n+1}, \dots, a^{i-1}) + \alpha |V|} \quad (6)$$

where $\text{cnt}(a^i, \dots, a^j)$ counts the number of occurrences of the action sub-sequence (a^i, \dots, a^j) in the training set, α is a smoothing constant, and V is the token vocabulary. Note that this model is trained in a *context-independent* way and only captures basic linguistic structure and common affordance relations observed in human actions. We optimize the parameters (n, α) to minimize the perplexity on a held-out validation set of actions.

To generate top actions given context c , we construct a restricted action space $\mathcal{A}_c = \mathcal{V} \times \mathcal{B}_c$, where \mathcal{V} is the set of verb phrases (e.g. *open*, *knock on*) collected from training actions, and \mathcal{B}_c is the set of nouns (e.g. *door*) detected in c using spaCy's[†] noun-phrase detection. Then we calculate $p_{(n,\alpha)}(a)$ for each $a \in \mathcal{A}_c$ and choose the top ones.

2. GPT-2 (Radford et al., 2019): We use a pre-trained GPT-2 and train it on \mathcal{D} according to (4) and (5). Unlike the previous n-gram model, GPT-2 helps model dependencies between the context and the action in a flexible way, relying on minimal assumptions about the structure of actions. We use beam search to generate most likely actions.

3.3 Reinforcement Learning with CALM

Though language models learn to generate useful actions, they are not optimized for gameplay performance. Therefore, we use CALM to generate top- k action candidates $A_{\text{LM}}(c, k) \subset A$ given context c , and train a DRRN to learn a Q-function over this action space. This can be done by simply replacing A with $A_{\text{LM}}(c, k)$ in equations (2) and (3). In this way, we combine the CALM's generic action priors with the ability of RL to learn policies optimized for the gameplay. We choose not to fine-tune CALM in RL so as to avoid overfitting to a specific game and invalidate the general priors present in CALM.

To summarize, we employ CALM for providing a reduced action space for text adventure agents to explore efficiently. Even though we choose a specific RL agent (DRRN) in our experiments, CALM is simple and generic, and can be combined with any RL agent.

4 Experimental Setup

We perform empirical studies to 1) evaluate the quality of actions generated by CALM in isolation from the complexities of RL, 2) evaluate CALM combined with an RL agent for gameplay, and 3) analyze what factors contribute to the effectiveness of our method. We describe our setup in this section and provide results in Section 5.

4.1 Data and Environment

ClubFloyd Dataset We collect data from ClubFloyd[‡], which archives transcripts of humans cooperatively playing text-based games. We

[†]<https://spacy.io/>

[‡]http://www.allthingsjacq.com/interactive_fiction.html#clubfloyd

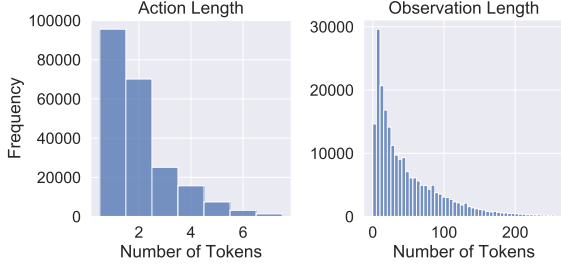


Figure 3: Distributions of actions and observations in the ClubFloyd Dataset, in terms of the number of tokens. Actions more than 7 tokens ($<0.5\%$) and observations more than 256 tokens ($<2\%$) are trimmed.

crawl 426 transcripts covering 590 games (in some transcripts people play more than one game), and build a dataset of 223,527 context-action pairs $\{((o_{t-1}, a_{t-1}, o_t), a_t)\}$. We pre-process the data by removing samples with meta-actions (e.g. ‘save’, ‘restore’) or observations with over 256 tokens. Figure 3 visualizes the action and observation length distributions. We also note that a few common actions (e.g. ‘north’, ‘take all’, ‘examine’) make up a large portion of the data. More details on the dataset are in the supplementary material.

Game Environment To test our RL agents, we use 28 man-made text games from the Jericho framework (Hausknecht et al., 2019a). We augment state observations with location and inventory descriptions by issuing the ‘look’ and ‘inventory’ commands, following the standard practice described in Hausknecht et al. (2019a).

The Jericho framework implements an *admissible action handicap* by enumerating all combinations of game verbs and objects at each state, and testing each action’s admissibility by accessing the underlying simulator states and load-and-save functions. As a result, the handicap runs no faster than a GPT-2 inference pass, and could in fact be unavailable for games outside Jericho. Jericho also provides an optimal *walkthrough* trajectory to win each game. Table 1 provides statistics of the ClubFloyd Dataset and the Jericho walkthroughs. We observe that ClubFloyd has a much larger vocabulary and a diverse set of games, which makes it ideal for training CALM. We utilize Jericho walkthroughs in our standalone evaluation of CALM in § 5.1.

4.2 CALM Setup

Training For training CALM (n-gram), we condition only on the current observation, i.e. $c_t = o_t$

	ClubFloyd Dataset	Jericho Walkthroughs
# unique games	590	28
Vocab size	39,670	9,623
Vocab size (game avg.)	2,363	1,037
Avg. trajectory length	360	98
Action Quality	Non-optimal	Optimal

Table 1: Statistics of the ClubFloyd Dataset and Jericho walkthrough trajectories.

instead of $c_t = (o_{t-1}, a_{t-1}, o_t)$, since o_{t-1} and a_{t-1} may contain irrelevant objects to the current state. We split the dataset into 90% training set and 10% validation set, and choose n and α based on the validation set perplexity. We find a bi-gram model $n = 2$, $\alpha = 0.00073$ works best, achieving a per-action perplexity of 863, 808 on the validation set and 17, 181 on the training set.

For CALM (GPT-2), we start with a 12-layer, 768-hidden, 12-head, 117M parameter GPT-2 model pre-trained on the WebText corpus (Radford et al., 2019). The implementation and pretrained weights of this model are obtained from Wolf et al. (2019). We then train it on the ClubFloyd transcripts for 3 epochs to minimize (4). We split the dataset into 90% training set and 10% validation set and we obtain a training loss of 0.25 and a validation loss of 1.98. Importantly, *both models are trained only on transcripts that do not overlap with the 28 Jericho games we evaluate on.*

Generating Top Actions For every unique state of each game, we generate the top $k = 30$ actions. For CALM (n-gram), we enumerate all actions in \mathcal{A}_c plus 13 one-word directional actions (e.g. ‘north’, ‘up’, ‘exit’). To encourage action diversity, at most 4 actions are generated for each object $b \in \mathcal{B}_c$. For CALM (GPT-2), we use beam search with a beam size of 40, and then choose the top 30 actions.

4.3 RL Agent Setup

Training We use DRRN (He et al., 2015) to estimate Q-Values over action candidates generated by CALM. Following Hausknecht et al. (2019a), we use a FastText model (Joulin et al., 2017) to predict the admissibility of an action based on the game’s textual response and filter out candidate actions that are found to be inadmissible. We train the DRRN asynchronously on 8 parallel instances of the game environment for 10^6 steps in total. Following Narasimhan et al. (2015), we use a separate

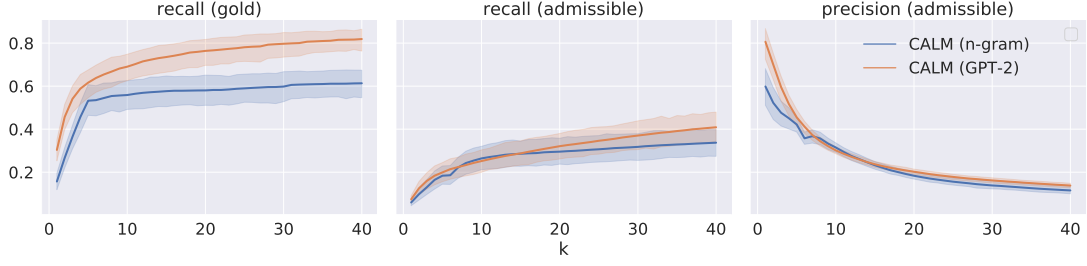


Figure 4: Precision and recall of gold and admissible actions generated by CALM, evaluated on walkthrough trajectories of 28 games provided by Jericho. k is the number of actions generated by CALM. Shaded areas represent standard deviation.

experience replay buffer to store trajectories with the best score at any point of time. The final score of a training run is taken to be the average score of the final 100 episodes during training. For each game, we train five independent agents with different random seeds and report the average score. For model variants in § 5.3 we only run one trail.

Baselines We compare with three baselines:

1. *NAIL* (Hausknecht et al., 2019b): Uses hand-written rules to act and explore, therefore requires no reinforcement learning or oracle access to admissible actions.

2. *DRRN* (He et al., 2015): This RL agent described in § 3.1 uses ground-truth admissible actions provided by the Jericho handicap.

3. *KG-A2C* (Ammanabrolu and Hausknecht, 2020): This RL agent constructs a game knowledge graph to augment the state space as well as constrain the types of actions generated. During learning, it requires the admissible action handicap to guide its exploration of the action space.

Of these methods, DRRN and KG-A2C require ground-truth admissible actions, which our model does not use, but we add them as reference comparisons for completeness.

5 Results

5.1 Evaluating CALM on walkthroughs

Metrics like validation loss or accuracy on validation set of our ClubFloyd data are not sufficient to evaluate CALM (see supplementary material for details on these metrics). This is because: 1) there can be multiple admissible actions in each state, and 2) the human actions in the trajectories are not guaranteed to be optimal or even admissible. Therefore, we use the walkthroughs provided in Jericho to provide an additional assessment on the quality of actions generated by CALM.

Consider a walkthrough to be an optimal trajectory $(o_1, a_1, \dots, o_l, a_l)$ leading to the maximum score achievable in the game. At step t ($1 \leq t \leq l$), the context c_t is (o_{t-1}, a_{t-1}, o_t) , the *gold* action is a_t and the full set of admissible actions A_t is obtained from the Jericho handicap. Suppose the generated set of top- k actions at step t is $A_{LM}(c_t, k)$. We then calculate the average precision of admissible actions ($prec_a$), recall of admissible actions (rec_a), and recall of gold actions (rec_g) as follows:

$$prec_a(k) = \frac{1}{l} \sum_{t=1}^l \frac{|A_t \cap A_{LM}(c_t, k)|}{k} \quad (7)$$

$$rec_a(k) = \frac{1}{l} \sum_{t=1}^l \frac{|A_t \cap A_{LM}(c_t, k)|}{|A_t|} \quad (8)$$

$$rec_g(k) = \frac{1}{l} \sum_{t=1}^l |\{a_t\} \cap A_{LM}(c_t, k)| \quad (9)$$

We calculate these metrics on each of the 28 games and present the averaged metrics as a function of k in Figure 4. The rec_a curve shows that the top $k = 15$ actions of CALM (GPT-2 and n-gram) are both expected to contain around 30% of all admissible actions in each walkthrough state. However, when k goes from 15 to 30, CALM (GPT-2) can come up with 10% more admissible actions, while the gains are limited for CALM (n-gram). When k is small, CALM (n-gram) benefits from its strong action assumption of one verb plus one object. However, this assumption also restricts CALM (n-gram) from generating more complex actions (e.g. ‘open case with key’) that CALM (GPT-2) can produce. This can also be seen in the rec_g curve, where the top-30 actions from CALM (GPT-2) contain the gold action in 20% more game states than CALM (n-gram). This gap is larger when it comes to gold actions, because they are more likely to be complex actions that the CALM (n-gram) is

Game	Without Handicap			With Handicap		Max Score
	CALM (GPT-2)	CALM (n-gram)	NAIL	KG-A2C	DRRN	
905	0	0	0	0	0	1
acorncourt	0	0	0	0.3	10	30
advland	0	0	0	0	20.6	100
advent	36	36	36	36	36	350
anchor	0	0	0	0	0	100
awaken	0	0	0	0	0	50
balances	9.1	8.9	10	10	10	51
deephome	1.0	1.0	13.3	1.0	1.0	300
<u>detective</u>	289.7	284.3	136.9	207.9	197.8	360
<u>dragon</u>	0.1	0.0	0.6	0	-3.5	25
enchanter	19.1	0	0	12.1	20	400
<u>inhumane</u>	25.7	1.7	0.6	3.0	0.7	90
jewel	0.3	0	1.6	1.8	1.6	90
karn	2.3	0	1.2	0	2.1	170
library	9.0	5.1	0.9	14.3	17.0	30
ludicorp	10.1	5.4	8.4	17.8	13.8	150
moonlit	0	0	0	0	0	1
omniquest	6.9	4.5	5.6	3.0	16.8	50
pentari	0	0	0	50.7	27.2	70
<u>snacktime</u>	19.4	0	0	0	9.7	50
sorcerer	6.2	5.0	5.0	5.8	20.8	400
spellbrkr	40	39.9	40	21.3	37.8	600
spirit	1.4	0.6	1.0	1.3	0.8	250
temple	0	0	7.3	7.6	7.9	35
zenon	0	0	0	3.9	0	20
zork1	30.4	24.8	10.3	34.0	32.6	350
zork3	0.5	0	1.8	0.0	0.5	7
ztuu	3.7	0	0	9.2	21.6	100
avg. norm	9.4%	5.5%	5.6%	10.8%	13.0%	

Table 2: Performance of our models (CALM (GPT-2) and CALM (n-gram)) compared to baselines (NAIL, KG-A2C, DRRN) on Jericho. We report raw scores for individual games as well as average normalized scores (avg.norm). *Advent* and *Deephome*’s initial scores are 1 and 36, respectively. Underlined games represent those where CALM outperforms handicap-assisted methods KGA2C and DRRN.

unable to model.

Further, we note that as k increases, the average quality of the actions decreases ($prec_a$ curve), while they contain more admissible actions (rec_a curve). Thus, k plays an important role in balancing exploration (more admissible actions) with exploitation (a larger ratio of admissible actions) for the RL agent, which we demonstrate empirically in § 5.3. We provide several examples of generated actions from both models in the supplementary material.

5.2 Evaluating gameplay on Jericho

We provide scores of our CALM-augmented DRRN agent on individual games in Table 2. To take into account different score scales across games, we consider both the raw score and the normalized score (raw score divided by maximum score), and only report the average normalized score across games.

Of the handicap-free models, CALM (n-gram)

Variant	avg. norm
<i>CALM (default)</i>	9.4%
CALM (20%)	8.1%
CALM (50%)	8.4%
CALM (w/ Jericho)	10.9%
CALM (w/o PT)	6.8%
CALM ($k = 10$)	5.6%
CALM ($k = 20$)	9.6%
CALM ($k = 40$)	9.2%
CALM (random agent)	1.8%

Table 3: Average normalized scores on Jericho for different variants of CALM (GPT-2). CALM (default) is the CALM (GPT-2) model used for results in Table 2.

achieves similar performance to NAIL, while CALM (GPT-2) outperforms CALM (n-gram) and NAIL by 4.4% and 3.8% on absolute normalized scores, respectively. Relatively, this represents almost a 69% improvement over NAIL. Figure 5 presents a game-wise comparison between CALM (GPT-2) and NAIL.

Surprisingly, even when compared to handicap-assisted models, CALM (GPT-2) performs quite well. On 8 out of 28 games (underlined in Table 2), CALM (GPT-2) outperforms both DRRN and KG-A2C despite the latter having access to ground-truth admissible actions. This improvement is especially impressive on games like DETECTIVE, INHUMANE and SNACKTIME, where our normalized score is higher by more than 20%. We hypothesize CALM excludes some non-useful admissible actions like “throw egg at sword” that humans never issue, which can speed up exploration. Also, it is possible that CALM sometimes discover admissible actions even the handicap cannot (due to the imperfection of state change detection).

5.3 Analysis

What Factors Contribute to Gameplay? We now analyze various components and design choices made in CALM (GPT-2). First, we investigate how much of the model’s performance is due to pre-training on text corpora as opposed to training on our ClubFloyd data. Then, we vary the number of actions (k) generated by the model. We also consider combining CALM with a random agent instead of RL. This leads us to the following variants:

1. **CALM (X%)**: These variants are trained with only X% of the transcripts from ClubFloyd. $X = 0$ is equivalent to using a pre-trained GPT-

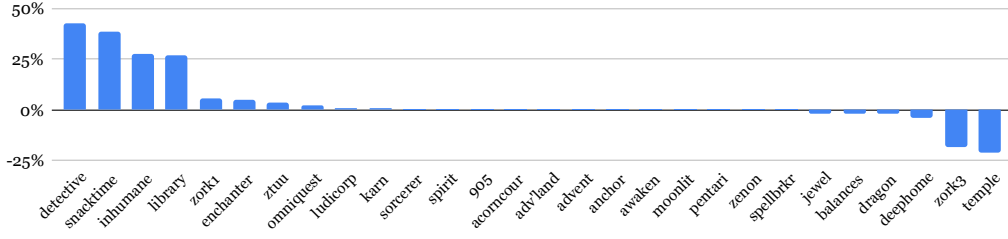


Figure 5: Difference in normalized scores achieved by CALM (GPT-2) and NAIL, in decreasing order.

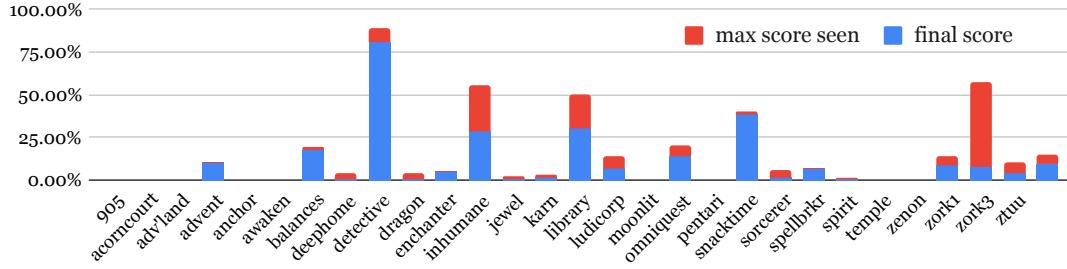


Figure 6: Final scores (blue) and maximum scores (normalized) seen during exploration (red) for CALM (GPT-2). There is a lot of potential for developing better algorithms to learn from high-scoring trajectories.

2 model off-the-shelf – we find that this fails to produce actions that are even parseable by the game engine and therefore is not reported in the table.

2. **CALM (w/ Jericho)**: This variant is trained on additional ClubFloyd data that includes 8 scripts from games contained in Jericho.

3. **CALM (w/o PT)**: This is a randomly initialized GPT-2 model, instead of a pre-trained one, trained on ClubFloyd data. We train this model for 10 epochs until the validation loss converges, unlike previous models which we train for 3 epochs.

4. **CALM ($k = Y$)**: This is a model variant that produces action sets of size Y .

5. **CALM (random agent)**: This model variant replaces DRRN by a random agent that samples uniformly from CALM top-30 actions at each state.

As shown in Table 3, the significant drop in score for CALM without pretraining shows that both pre-training and ClubFloyd training are important for gameplay performance. Pre-training provides general linguistic priors that regularize action generation while the ClubFloyd data conditions the model towards generating actions useful in text-based games.

Adding heldout transcripts from Jericho evaluation games (CALM w/ Jericho) provides additional benefit as expected, even surpassing handicap-assisted KG-A2C in terms of the average normalized score. Counter-intuitively, we find that the greatest performance gains aren’t on games fea-

tured in the heldout transcripts. See supplementary material for more details.

For the models with different k values, CALM ($k = 10$) is much worse than other choices, but similar to CALM (n-gram) in Table 2. Note that in Figure 4 the recall of admissible actions is similar between GPT-2 and n-gram when $k \leq 10$. We believe it is because top-10 GPT-2 actions are usually simple actions that occur a lot in ClubFloyd (e.g. ‘east’, ‘get object’), which is also what n-gram can capture. It is really the complex actions captured when $k > 10$ that makes GPT-2 much better than n-gram. On the other hand, though $k = 20, 30, 40$ achieve similar overall performance, they achieve different results for different games. So potentially the CALM overall performance can be further improved by choosing different k for different games. Finally, CALM (random agent) performs a poor score of 1.8%, and clearly shows the importance of combining CALM with an RL agent to adaptively choose actions.

Is CALM limiting RL? A natural question to ask is whether reducing the action space using CALM results in missing key actions that may have led to higher scores in the games. To answer this, we also plot the maximum scores seen by our CALM (GPT-2) agent during RL in Figure 6. Some games (e.g. 905, ACORNCOURT) are intrinsically hard to achieve any score. However, on other games with non-zero scores, DRRN is

unable to stably converge to the maximum score seen in RL exploration. If RL can fully exploit and learn from the trajectories experienced under the CALM action space for each game, the average normalized score would be 14.7%, higher than any model in Table 2, both with and without handicaps.

6 Conclusion

In this paper, we proposed the Contextual Action Language Model (CALM), a language model approach to generating action candidates for reinforcement learning agents in text-based games. Our key insight is to use language models to capture linguistic priors and game sense from humans game-play on a diverse set of games. We demonstrated that CALM can generate high-quality, contextually-relevant actions even for games unseen in its training set, and when paired with a DRRN agent, outperforms previous approaches on the Jericho benchmark (Hausknecht et al., 2019a) by as much as 69% in terms of average normalized score. Remarkably, on many of these games, our approach is competitive even with models that use ground truth admissible actions, implying that CALM is able to generate high-quality actions across diverse games and contexts.

From the results in Table 2, it is safe to conclude that text-based games are still far from being solved. Even with access to ground truth admissible actions, sparse rewards and partial observability pose daunting challenges for current agents. In the future, we believe that strong linguistic priors will continue to be a key ingredient for building next-level learning agents in these games. By releasing our dataset and code we hope to provide a solid foundation to accelerate work in this direction.

Acknowledgement

Gracious thanks to Jacqueline Ashwell for running ClubFloyd and agreeing to our use of the collected transcripts. We thank Danqi Chen, Jimmy Yang, Jens Tuyls, and other colleagues from Princeton NLP group for proofreading and discussion. We also thank reviewers for constructive feedbacks. This research was partially funded by the Center for Statistics and Machine Learning at Princeton University through support from Microsoft.

References

- Prithviraj Ammanabrolu, William Broniec, Alex Mueller, Jeremy Paul, and Mark O Riedl. 2019. Toward automated quest generation in text-adventure games. *arXiv preprint arXiv:1909.06283*.
- Prithviraj Ammanabrolu and Matthew Hausknecht. 2020. Graph constrained reinforcement learning for natural language action spaces. *arXiv preprint arXiv:2001.08837*.
- Prithviraj Ammanabrolu and Mark Riedl. 2019. Playing text-adventure games with graph-based deep reinforcement learning. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3557–3565.
- Hongshen Chen, Xiaorui Liu, Dawei Yin, and Jiliang Tang. 2017. A survey on dialogue systems: Recent advances and new frontiers. *Acm Sigkdd Explorations Newsletter*, 19(2):25–35.
- Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar. Association for Computational Linguistics.
- Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, et al. 2018. Textworld: A learning environment for text-based games. In *Workshop on Computer Games*, pages 41–75. Springer.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- Nancy Fulda, Daniel Ricks, Ben Murdoch, and David Wingate. 2017. What can you do with a rock? affordance extraction via word embeddings. *CoRR*, abs/1703.03429.
- Matthew Hausknecht, Prithviraj Ammanabrolu, Marc-Alexandre Côté, and Xingdi Yuan. 2019a. Interactive fiction games: A colossal adventure. *CoRR*, abs/1909.05398.
- Matthew Hausknecht, Ricky Loynd, Greg Yang, Adith Swaminathan, and Jason D Williams. 2019b. Nail: A general interactive fiction agent. *arXiv preprint arXiv:1902.04259*.
- Ji He, Jianshu Chen, Xiaodong He, Jianfeng Gao, Li-hong Li, Li Deng, and Mari Ostendorf. 2015. Deep reinforcement learning with a natural language action space. *arXiv preprint arXiv:1511.04636*.

- Vishal Jain, William Fedus, Hugo Larochelle, Doina Precup, and Marc G. Bellemare. 2019. Algorithmic improvements for deep reinforcement learning applied to interactive fiction. *CoRR*, abs/1903.03094.
- Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. 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*, pages 427–431. Association for Computational Linguistics.
- Daniel Jurafsky and James H. Martin. 2009. *Speech and Language Processing (2nd Edition)*. Prentice-Hall, Inc., USA.
- Bartosz Kostka, Jaroslaw Kwiecien, Jakub Kowalski, and Pawel Rychlikowski. 2017. Text-based adventures of the golovin AI agent. *CoRR*, abs/1705.05637.
- Angeliki Lazaridou, Anna Potapenko, and Olivier Tieleman. 2020. Multi-agent communication meets natural language: Synergies between functional and structural language learning. *arXiv preprint arXiv:2005.07064*.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.
- Karthik Narasimhan, Tejas D. Kulkarni, and Regina Barzilay. 2015. Language understanding for text-based games using deep reinforcement learning. In *EMNLP*, pages 1–11.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.
- Yiping Song, Rui Yan, Xiang Li, Dongyan Zhao, and Ming Zhang. 2016. Two are better than one: An ensemble of retrieval-and generation-based dialog systems. *arXiv preprint arXiv:1610.07149*.
- Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.
- Ruo Yu Tao, Marc-Alexandre Côté, Xingdi Yuan, and Layla El Asri. 2018. Towards solving text-based games by producing adaptive action spaces. *arXiv preprint arXiv:1812.00855*.
- Jack Urbanek, Angela Fan, Siddharth Karamcheti, Saachi Jain, Samuel Humeau, Emily Dinan, Tim Rocktäschel, Douwe Kiela, Arthur Szlam, and Jason Weston. 2019. Learning to speak and act in a fantasy text adventure game. *CoRR*, abs/1903.03094.
- Nick Walton. 2019. Ai dungeon 2: Creating infinitely generated text adventures with deep learning language models.
- Jason D. Williams, Kavosh Asadi, and Geoffrey Zweig. 2017. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 665–677, Vancouver, Canada. Association for Computational Linguistics.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R’emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface’s transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.
- Tom Zahavy, Matan Haroush, Nadav Merlis, Daniel J Mankowitz, and Shie Mannor. 2018. Learn what not to learn: Action elimination with deep reinforcement learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 3562–3573. Curran Associates, Inc.
- Tiancheng Zhao and Maxine Eskenazi. 2016. Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 1–10, Los Angeles. Association for Computational Linguistics.

A ClubFloyd Dataset

The ClubFloyd transcripts we collected are game-play logs generated by a group of people that regularly meet to play interactive fiction games. The participants are experienced at playing text-based games, however they may not be familiar with the game that’s being played, and do make several mistakes. We include a snippet of a transcript in Figure 7. We crawled the ClubFloyd website to acquire 426 transcripts, spanning over 500 games.

To process a transcript, we clean the data and extract observations and actions. The data contains several sources of noise, which we remove: the first is non-game information such as chat logs between the humans playing the games; second are meta-actions that humans use to save and load games and navigate menus; and finally, we remove typos, expand common abbreviations (“n” to “north”, “x” to “examine”, etc.), and filter out any actions that weren’t recognized by the game parsers.

Once we have our cleaned observations and actions, we group observations and actions into the

Jacqueline says (to Floyd), "s"

Floyd | You'll have to get out of the car first.
Floyd |
Floyd | >

Jacqueline says (to Floyd), "put car in reverse"

Floyd | [That object is either not here or not important.] Gunther asks, "'drive'?"
Floyd |
Floyd | >

Jacqueline says (to Floyd), "drive"

Floyd | (the car) eladnarra says, "howdy"
Floyd |
Floyd | Driving
Floyd | Ah, scenic Las Mesas. Man, this place is an absolute toilet.
Soon
Floyd | you'll be able to afford to get the hell out of here -- provided
you
Floyd | can avoid making any more slip-ups on the job.
Floyd |
Floyd | As you cruise down the road, you notice a freeway onramp
approaching.
Floyd | Would you like to get on? >>

Jacqueline says, "Oh."

Jacqueline says, "heh"

eladnarra says, "ohey, a game
I've actually played"

Jacqueline says, "So, I have no
idea where I'm going, then."

Jacqueline says, "Awesome."

Jacqueline says, "Sure, let's
take the on ramp."

Jacqueline says (to Floyd), "yes"

Figure 7: Selection from a raw ClubFloyd Transcript of the game 9:05

<p>[OBS] [That object is either not here or not important.] [ACTION] south [OBS] You'll have to get out of the car first. [ACTION] put car in reverse</p> <p>[OBS] You'll have to get out of the car first. [ACTION] put car in reverse [OBS] [That object is either not here or not important.] [ACTION] drive</p> <p>[OBS] [That object is either not here or not important.] [ACTION] drive [OBS] (the car) Driving Ah, scenic Las Mesas. Man, this place is an absolute toilet. Soon you'll be able to afford to get the hell out of here -- provided you can avoid making any more slip-ups on the job. As you cruise down the road, you notice a freeway onramp approaching. Would you like to get on? >> [ACTION] yes</p>

Figure 8: Cleaned section of Figure 7

form $(o_{j-1}, a_{j-1}, o_j), a_j$. For the very first observation and action, we pad the beginning of the example with the observation "You are at the start of your journey" and the action "begin journey".

After this entire pre-processing, the dataset contains 223,527 examples.

B CALM Training

In this section, we will provide training details of CALM (GPT-2), CALM (n-gram), and their variants.

B.1 CALM (GPT-2)

We first discuss the CALM (GPT-2) models, and begin with the portion of the ClubFloyd data that they are trained on. We begin with a 12-layer, 768-hidden, 12-head, 117M parameter pretrained OpenAI GPT-2 model.

We note that the number of samples we train on, even in the CALM (GPT-2) model + Jericho games variant, is less than the total samples in the dataset. This is because we do not train on incomplete batches of data, and we omit samples that exceed 256 tokens.

CALM (GPT-2) To train CALM (GPT-2), we take transcripts from ClubFloyd (excluding Jericho games) and order the samples based on the transcript number they came from. This yields a dataset of 193,588 samples. We select the first 90% of the samples as train data, and the last 10% of the samples as validation data.

CALM (GPT-2) 50%, 20%, (+) Jericho To train the 50% and 20% variants, we select without replacement 212 transcripts (94,609 samples), and 85 transcripts (38,334 samples) respectively from the ClubFloyd transcripts (excluding Jericho games). We order the samples based on the transcript they come from, choose the first 90% of the data as our training data and last 10% as validation data.

For the CALM (GPT-2) variant including Jericho games, we include every ClubFloyd transcript, we randomly order the transcripts, order the samples based on the order of the transcripts, and then we select the first 90% of the data as our training data, and the last 10% of the data as validation data. This split contains 206,286 samples.

CALM (GPT-2) Random Initialization For the CALM (GPT-2) variant with random initialization, we begin with a GPT-2 model that has not been pre-trained. We only use the transcripts in ClubFloyd that do not correspond to any Jericho game we test on. We randomly order the transcripts, and order the samples based on the order of the transcripts. We select the first 90% of the data as our training data, and the last 10% of the data as validation data.

Parameter Optimization In order to train GPT-2, we minimize the cross-entropy between GPT-2's distribution over actions and the action taken in the ClubFloyd example. We use Adam to optimize the weights of our model with learning rate = $2e-5$ and Adam epsilon = $1e-8$. For the learning rate we use a linear schedule with warmup. Finally, we clip gradients allowing a max gradient norm of 1.

We include the loss on the train and validation set, as well as the accuracy (defined as the percentage of examples on which the action assigned the

Model	Metric	1	2	3	4	5	9	10
Main	Train Loss	0.32	0.27	0.25	0.23	0.22	n/a	n/a
	Train Acc	0.11	0.14	0.16	0.18	0.19	n/a	n/a
	Val Loss	2.14	2.04	1.98	1.96	1.96	n/a	n/a
	Val Acc	0.13	0.15	0.16	0.17	0.18	n/a	n/a
50%	Train Loss	0.66	0.55	0.49	0.46	0.43	n/a	n/a
	Train Acc	0.11	0.14	0.17	0.19	0.21	n/a	n/a
	Val Loss	2.19	2.09	2.06	2.04	2.05	n/a	n/a
	Val Acc	0.14	0.15	0.15	0.16	0.16	n/a	n/a
20%	Train Loss	0.37	0.29	0.26	0.25	0.24	n/a	n/a
	Train Acc	0.08	0.11	0.13	0.15	0.16	n/a	n/a
	Val Loss	2.32	2.17	2.12	2.09	2.08	n/a	n/a
	Val Acc	0.10	0.12	0.13	0.14	0.15	n/a	n/a
Jericho	Train Loss	0.62	0.53	0.48	0.45	0.43	n/a	n/a
	Train Acc	0.12	0.16	0.19	0.21	0.23	n/a	n/a
	Val Loss	2.10	2.00	1.97	1.96	1.98	n/a	n/a
	Val Acc	0.16	0.17	0.17	0.18	0.18	n/a	n/a
Random Init	Train Loss	0.36	0.33	0.31	0.29	0.27	0.23	0.23
	Train Acc	0.05	0.07	0.08	0.10	0.11	0.15	0.15
	Val Loss	4.96	4.60	4.35	4.16	4.01	3.73	3.73
	Val Acc	0.06	0.08	0.09	0.10	0.10	0.12	0.12

Table 4: Training Metrics for CALM Variants

highest probability by GPT-2 was the ClubFloyd action) in Table 4.

B.2 CALM (n-gram)

In order to train the CALM n-gram model, we consider the set of transcripts in ClubFloyd (excluding Jericho games). Next, we take the set of actions that appear in these transcripts, and train an n-gram model with Laplace α smoothing to model these sequences (Jurafsky and Martin, 2009). We order actions by the transcript they appear in and take the first 70% of the actions as train data and leave the remaining 30% as validation data. For each n , we choose alpha that minimizes perplexity per word on the validation data. We also tried a linear interpolation of these estimates (Jurafsky and Martin, 2009) although we did not observe an improvement over our bigram model. In this model, we estimate $p(a^i|a^{i-3}, a^{i-2}, a^{i-1}) = w_1 p^*(a^i|a^{i-3}, a^{i-2}, a^{i-1}) + w_2 p^*(a^i|a^{i-2}, a^{i-1}) + w_3 p^*(a^i|a^{i-1}) + w_4 p^*(a^i)$ where $\sum_i w_i = 1$, and p^* indicates our m-gram estimate for $p(a^i|a^{i-m+1}, \dots, a^{i-1})$.

C Walkthrough Evaluation

In Figure 10, we provide a piece of walkthrough trajectory of Zork1, with GPT-2 and n-gram generated actions at each state. Note that n-gram actions are mostly limited to be no more than two tokens, while GPT-2 can generate more complex actions like “put sword in case”.

In Figure 9, we provide game-specific metric curves for Zork1 and Detective. On harder games like Zork1, there is significant gap between GPT-2 and n-gram, while easy games like Detective the gap is very small.

D Gameplay Evaluation

On Zork1, we provide learning curves for CALM (GPT-2) (Figure 11) and CALM (n-gram) (Figure 12). We also provide trail curves for CALM (GPT-2) on Zork3 (Figure 14), a game we are behind NAIL, and trails using different top- $k \in \{10, 20, 30, 40\}$ actions by CALM (GPT-2) on Zork1 (Figure 13).

We provide per-game results for model variants in Table 5. It is interesting that CALM (w/ Jericho) is significantly better than CALM (GPT-2) on the games of Temple and Deephome (non-trivial scores achieved), which are not the games with ClubFloyd scripts added. On the other hand, games like 905 and moonlit have scripts added, but do not get improved.

In the end, we append one example trajectory piece of DRRN + CALM (GPT-2) on Zork1 (Figure 15), where CALM generated action candidates and their Q-values are shown along with observations, actions and scores.

Game	CALM (GPT-2)	CALM (ngram)	CALM (w/o PT)	CALM (20%)	CALM (50%)	CALM (w/ Jericho)	CALM (k=10)	CALM (k=20)	CALM (k=40)	CALM (random agent)	Max Score
905	0.00 (± 0.00)	0.00 (± 0.00)	0.00	0.00	0.00	0.00 (± 0.00)	0.00	0.00	0.00	0.00	1
acorncourt	0.00 (± 0.00)	0.00 (± 0.00)	0.00	0.00	0.00	0.00 (± 0.00)	0.00	0.00	0.00	0.00	30
adv'land	0.00 (± 0.00)	0.00 (± 0.00)	0.00	0.00	0.00	0.00 (± 0.00)	0.00	0.00	0.00	0.00	100
advent	36.00 (± 0.00)	36.00 (± 0.00)	36.00	36.00	36.00	36.00 (± 0.00)	36.00	36.00	36.00	36.00	350
anchor	0.00 (± 0.00)	0.00 (± 0.00)	0.00	0.00	0.00	0.00 (± 0.00)	0.00	0.00	0.00	0.00	100
awaken	0.00 (± 0.00)	0.00 (± 0.00)	0.00	0.00	0.00	0.00 (± 0.00)	0.00	0.00	0.00	0.00	50
balances	9.15 (± 0.08)	8.86 (± 0.04)	6.00	7.89	9.43	4.05 (± 0.15)	0.00	9.17	8.07	1.70	51
deephome	1.00 (± 0.00)	1.00 (± 0.00)	1.00	1.00	1.00	6.95 (± 5.43)	1.00	1.00	1.00	1.05	300
detective	289.71 (± 0.20)	284.33 (± 11.04)	288.21	289.30	289.58	289.87 (± 0.11)	289.75	289.51	290.04	40.00	360
dragon	0.13 (± 0.05)	0.05 (± 0.03)	0.00	0.27	0.25	0.19 (± 0.03)	0.32	0.12	0.18	-0.19	25
enchanter	19.09 (± 0.59)	0.00 (± 0.00)	0.00	0.00	0.00	19.92 (± 0.06)	0.00	15.33	20.00	0.00	400
inhumane	25.73 (± 2.93)	1.72 (± 0.93)	0.00	20.15	22.38	28.16 (± 3.32)	8.38	30.03	21.73	0.00	90
jewel	0.27 (± 0.01)	0.00 (± 0.00)	0.00	0.00	0.00	0.38 (± 0.05)	0.00	0.20	0.46	0.00	90
karn	2.30 (± 0.05)	0.00 (± 0.00)	0.00	3.19	1.73	2.19 (± 0.08)	0.14	2.63	1.71	0.00	170
library	9.02 (± 5.07)	5.07 (± 0.28)	13.77	12.31	11.84	12.47 (± 0.35)	3.22	10.40	10.46	1.74	30
ludicorp	10.09 (± 0.60)	5.44 (± 0.04)	11.39	11.40	9.87	10.64 (± 0.90)	10.93	11.72	9.00	6.72	150
moonlit	0.00 (± 0.00)	0.00 (± 0.00)	0.00	0.00	0.00	0.00 (± 0.00)	0.00	0.00	0.00	0.00	1
omniquest	6.88 (± 0.10)	4.53 (± 0.09)	4.80	7.08	5.79	6.87 (± 0.15)	4.98	6.20	6.55	3.10	50
pentari	0.00 (± 0.00)	0.00 (± 0.00)	0.00	0.00	0.00	0.00 (± 0.00)	0.00	0.00	0.00	0.00	70
snacktime	19.40 (± 0.29)	0.00 (± 0.00)	0.00	0.00	7.84	31.75 (± 8.62)	0.00	19.25	20.14	0.50	50
sorcerer	6.18 (± 1.80)	5.00 (± 0.00)	5.00	5.03	5.73	5.65 (± 1.45)	11.57	5.00	5.00	5.00	400
spellbrkr	39.99 (± 0.01)	39.92 (± 0.03)	39.94	39.97	39.86	40.00 (± 0.00)	40.00	39.96	40.00	36.20	600
spirit	1.36 (± 0.03)	0.64 (± 0.07)	1.78	1.23	1.32	1.23 (± 0.05)	1.85	1.51	1.21	0.20	250
temple	0.00 (± 0.00)	0.00 (± 0.00)	0.00	0.00	0.00	3.52 (± 1.99)	0.00	0.00	0.00	0.00	35
zenon	0.00 (± 0.00)	0.00 (± 0.00)	0.00	0.00	0.00	0.00 (± 0.00)	0.00	0.00	0.00	0.00	20
zork1	30.39 (± 3.01)	24.76 (± 0.52)	11.30	22.75	27.44	32.17 (± 4.39)	12.70	31.36	29.10	2.40	350
zork3	0.53 (± 0.08)	0.02 (± 0.01)	0.89	0.79	0.34	0.46 (± 0.06)	0.97	0.49	0.26	0.07	7
ztuu	3.74 (± 0.30)	0.00 (± 0.00)	0.00	5.66	4.85	3.93 (± 0.07)	0.00	3.73	4.38	0.55	100

Table 5: Raw scores for variants of CALM (GPT-2) on each game. Games in bold are those with ClubFloyd scripts. Note that some scores are only based on one trial. CALM (GPT-2), CALM (ngram) and CALM (w/ Jericho) are based on five trails and the standard deviation is given.

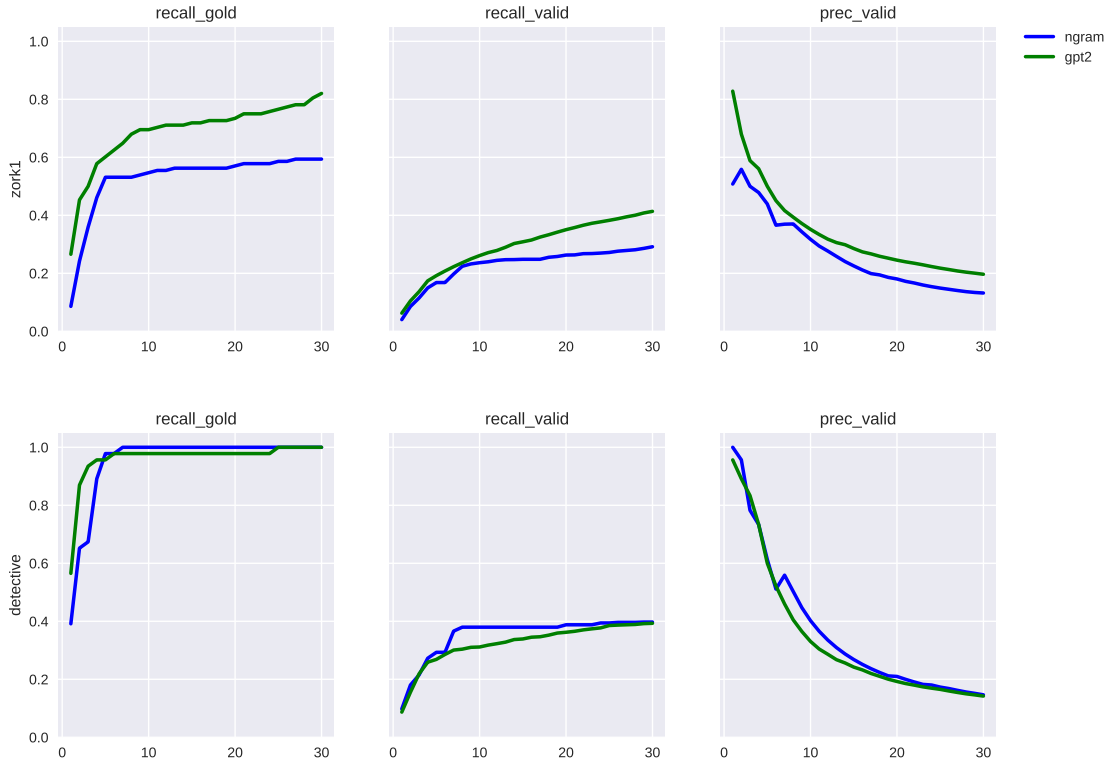


Figure 9: Walkthrough evaluation for Zork1 and Detective.

```

step 22
state: [CLS] living room above the trophy case hangs an elvish sword of great antiquity. [SEP] get sword [SEP] taken. you
are carrying: a sword a nasty knife a rope a brass lantern a clove of garlic a jewel-encrusted egg living
room you are in the living room. there is a doorway to the east, a wooden door with strange gothic lettering to the west
, which appears to be nailed shut, a trophy case, and a large oriental rug in the center of the room. [SEP]
gpt2_acts: ['east', 'west', 'north', 'south', 'up', 'down', 'drop sword', 'open trophy case', 'open door', 'southeast', 'wait',
'southwest', 'northwest', 'wear sword', 'northeast', 'out', 'take sword', 'knock on door', 'get statue', 'open
trophy', 'get rug', 'close door', 'take rug', 'get all', 'get sword', 'open case', 'take all', 'put sword in case', '
get trophy', 'open gothic']
ngram_acts: ['north', 'east', 'south', 'west', 'open door', 'examine door', 'take all', 'unlock door', 'get all', 'close door',
'drop all', 'put all', 'tie rope', 'examine knife', 'take knife', 'examine case', 'examine sword', 'open case', '
examine rug', 'examine rope', 'examine west', 'take rope', 'take sword', 'examine lantern', 'put knife', 'pull rope', '
take lantern', 'examine egg', 'take rug', 'look under rug']
valid_acts: ['east', 'open egg with lantern', 'throw rope at egg', 'throw egg at knife', 'throw sword at egg', 'throw garlic
at egg', 'throw lantern at egg', 'throw knife at egg', 'throw knife at lantern', 'push rug', 'put down all', 'put down
rope', 'put down egg', 'put down sword', 'put down garlic', 'put down lantern', 'put down knife', 'take on egg', 'open
case', 'turn on lantern']
gold_act: ['push rug']
score: 15

step 23
state: [CLS] taken. [SEP] move rug [SEP] with a great effort, the rug is moved to one side of the room, revealing the dusty
cover of a closed trap door. you are carrying: a sword a nasty knife a rope a brass lantern a clove of garlic
a jewel-encrusted egg living room you are in the living room. there is a doorway to the east, a wooden door with
strange gothic lettering to the west, which appears to be nailed shut, a trophy case, and a closed trap door at your
feet. [SEP]
gpt2_acts: ['east', 'west', 'open door', 'north', 'open trap door', 'south', 'open case', 'down', 'up', 'wait', 'open trap',
'pull rug', 'move cover', 'knock on door', 'southeast', 'push rug', 'search cover', 'open cover', 'out', 'close trap',
'southwest', 'move rug', 'enter trap', 'open gothic', 'drop sword', 'search rug', 'northwest', 'close trap door', 'take
rug', 'take all']
ngram_acts: ['north', 'east', 'south', 'west', 'open door', 'examine door', 'take all', 'unlock door', 'get all', 'close door',
'drop all', 'put all', 'tie rope', 'examine knife', 'take knife', 'examine case', 'examine sword', 'open case', '
examine rug', 'examine rope', 'examine west', 'take rope', 'take sword', 'examine lantern', 'put knife', 'pull rope', '
take lantern', 'examine egg', 'take rug', 'look under rug']
valid_acts: ['east', 'open egg with lantern', 'throw rope at egg', 'throw egg at knife', 'throw sword at egg', 'throw garlic
at egg', 'throw lantern at egg', 'throw knife at egg', 'throw knife at lantern', 'put down all', 'put down rope', 'put
down egg', 'put down sword', 'put down garlic', 'put down lantern', 'put down knife', 'take on egg', 'open trap', 'open
case', 'turn on lantern']
gold_act: ['open trap']
score: 15

step 24
state: [CLS] with a great effort, the rug is moved to one side of the room, revealing the dusty cover of a closed trap door.
[SEP] open trapdoor [SEP] the door reluctantly opens to reveal a rickety staircase descending into darkness. you are
carrying: a sword a nasty knife a rope a brass lantern a clove of garlic a jewel-encrusted egg living room
you are in the living room. there is a doorway to the east, a wooden door with strange gothic lettering to the west,
which appears to be nailed shut, a trophy case, and a rug lying beside an open trap door. [SEP]
gpt2_acts: ['east', 'west', 'down', 'up', 'north', 'south', 'open trophy case', 'wait', 'knock on door', 'take rug', '
southeast', 'enter trapdoor', 'out', 'drop sword', 'take rope', 'in', 'southwest', 'northwest', 'get rope', 'open case',
'get rug', 'search rug', 'enter trap', 'climb rope', 'northeast', 'take sword', 'move rug', 'take all', 'put sword
in trapdoor', 'close trapdoor']
ngram_acts: ['north', 'east', 'south', 'west', 'open door', 'examine door', 'take all', 'unlock door', 'get all', 'close door',
'drop all', 'put all', 'tie rope', 'examine knife', 'take knife', 'examine case', 'examine sword', 'open case', '
examine rope', 'examine west', 'take rope', 'take sword', 'examine lantern', 'put knife', 'pull rope', 'take lantern',
'examine egg', 'put sword', 'get sword', 'put egg']
valid_acts: ['east', 'open egg with lantern', 'throw rope at egg', 'throw egg at knife', 'throw sword at egg', 'throw garlic
at egg', 'throw lantern at egg', 'throw knife at egg', 'throw knife at lantern', 'put down all', 'put down rope', 'put
down egg', 'put down sword', 'put down garlic', 'put down lantern', 'put down knife', 'close trap', 'take on egg', '
open case', 'turn on lantern', 'down']
gold_act: ['down']
score: 15

step 25
state: [CLS] the door reluctantly opens to reveal a rickety staircase descending into darkness. [SEP] down [SEP] you have
moved into a dark place. the trap door crashes shut, and you hear someone barring it. it is pitch black. you are likely
to be eaten by a grue. your sword is glowing with a faint blue glow. you are carrying: a sword a nasty knife a
rope a brass lantern a clove of garlic a jewel-encrusted egg it is pitch black. you are likely to be eaten by a
grue. [SEP]
gpt2_acts: ['down', 'west', 'east', 'north', 'wait', 'south', 'up', 'open door', 'southeast', 'listen', 'southwest', 'out', '
northeast', 'open trap door', 'northwest', 'enter trap', 'drop sword', 'sleep', 'close door', 'knock on door', 'get
rope', 'open trap', 'turn off lamp', 'sing', 'stand', 'take rope', 'forward', 'shout', 'pull rope', 'sound']
ngram_acts: ['north', 'east', 'south', 'west', 'open door', 'examine door', 'take all', 'unlock door', 'get all', 'close door',
'drop all', 'put all', 'tie rope', 'examine knife', 'take knife', 'examine case', 'examine sword', 'open case', '
examine rope', 'examine west', 'take rope', 'take sword', 'examine lantern', 'put knife', 'pull rope', 'take lantern',
'examine egg', 'put sword', 'get sword', 'put egg']
valid_acts: ['south', 'north', 'open egg with lantern', 'throw rope at egg', 'throw egg at sword', 'throw garlic at egg', '
throw lantern at egg', 'throw knife at egg', 'throw sword at egg', 'throw sword at lantern', 'put down all', 'put down
rope', 'put down egg', 'put down garlic', 'put down lantern', 'put down knife', 'put down sword', 'take on egg', 'turn
on lantern', 'east']
gold_act: ['turn on lantern']
score: 40

```

Figure 10: A piece of walkthrough evaluation in Zork1.

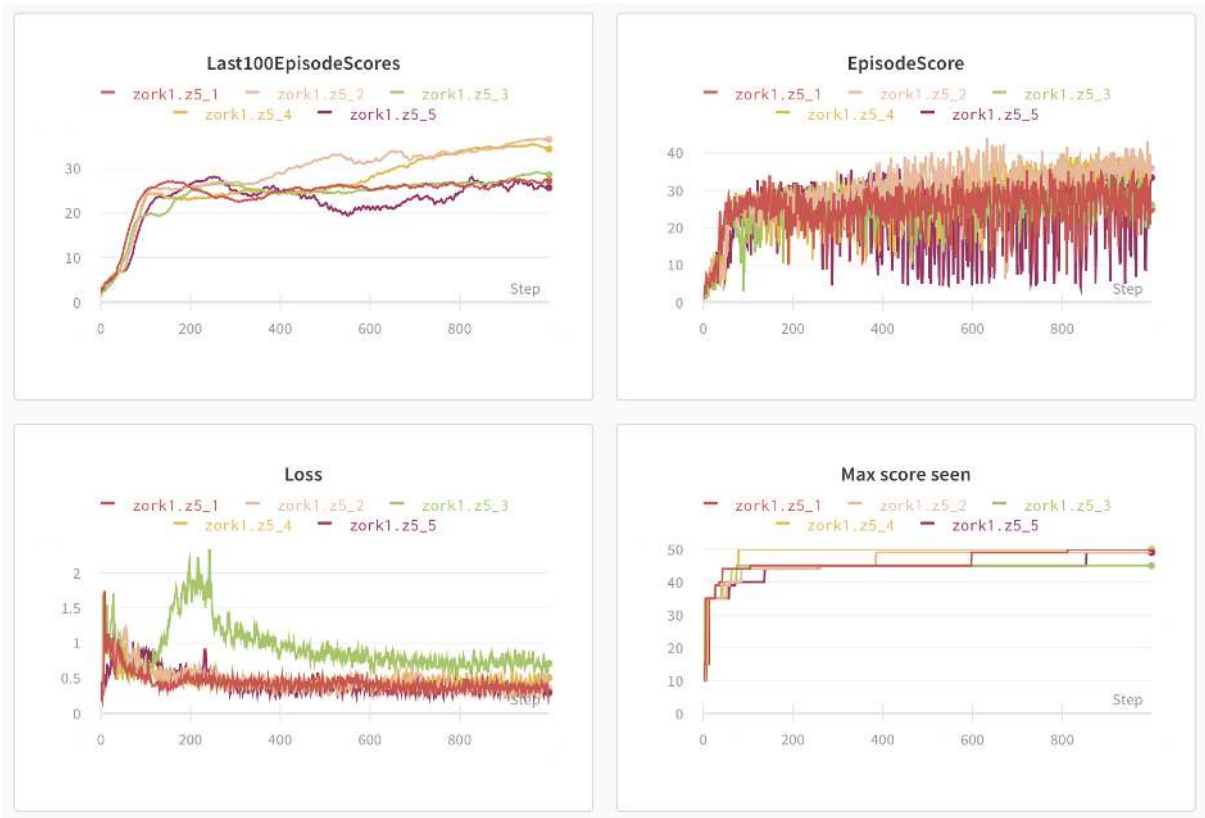


Figure 11: CALM (GPT-2) learning Zork1. Results show the five independent training runs.

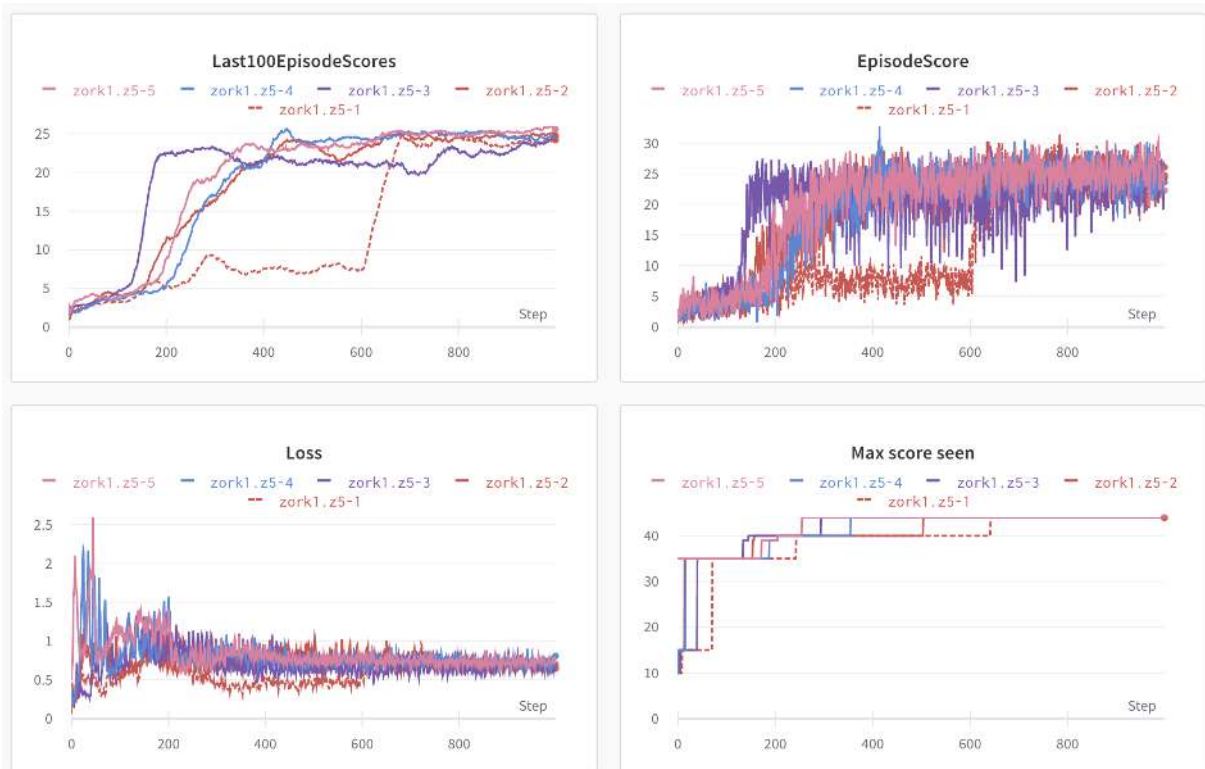


Figure 12: CALM (n-gram) learning Zork1. Results show the five independent training runs.

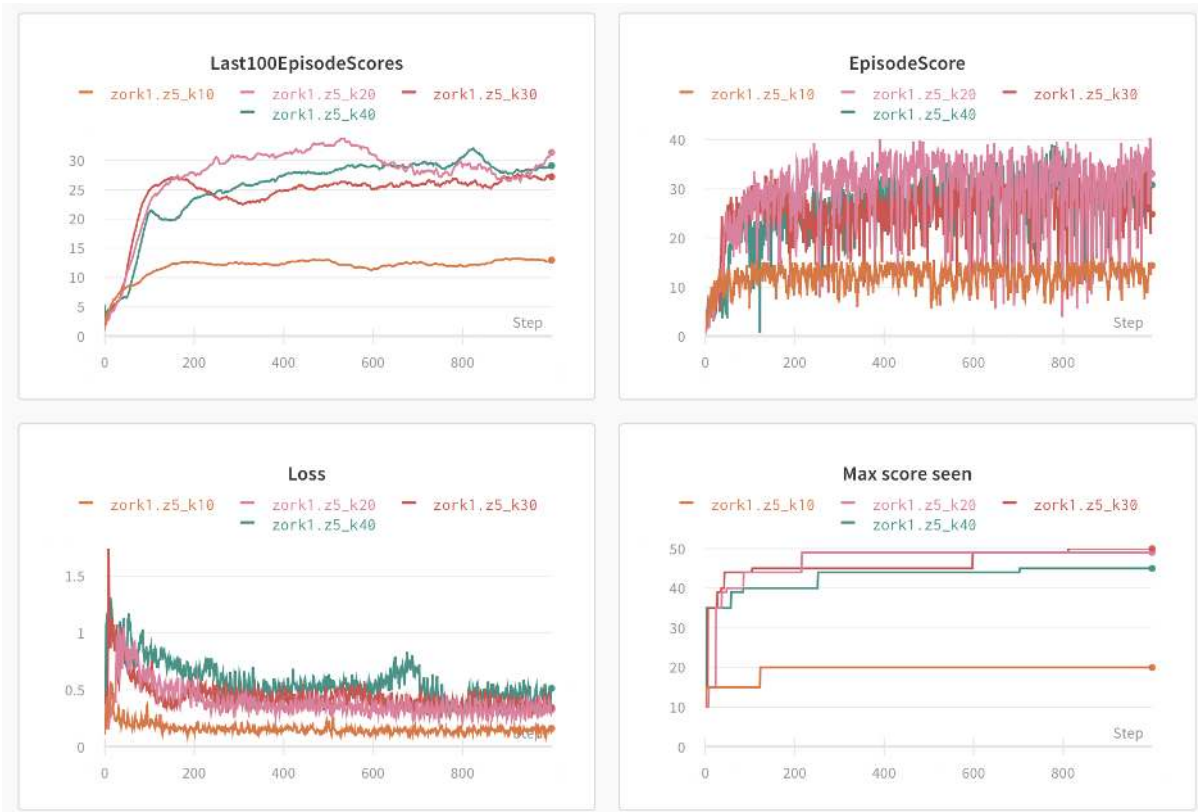


Figure 13: CALM (GPT-2) on Zork1 when decoding variable numbers of top- k actions ($k = 10, 20, 30, 40$).

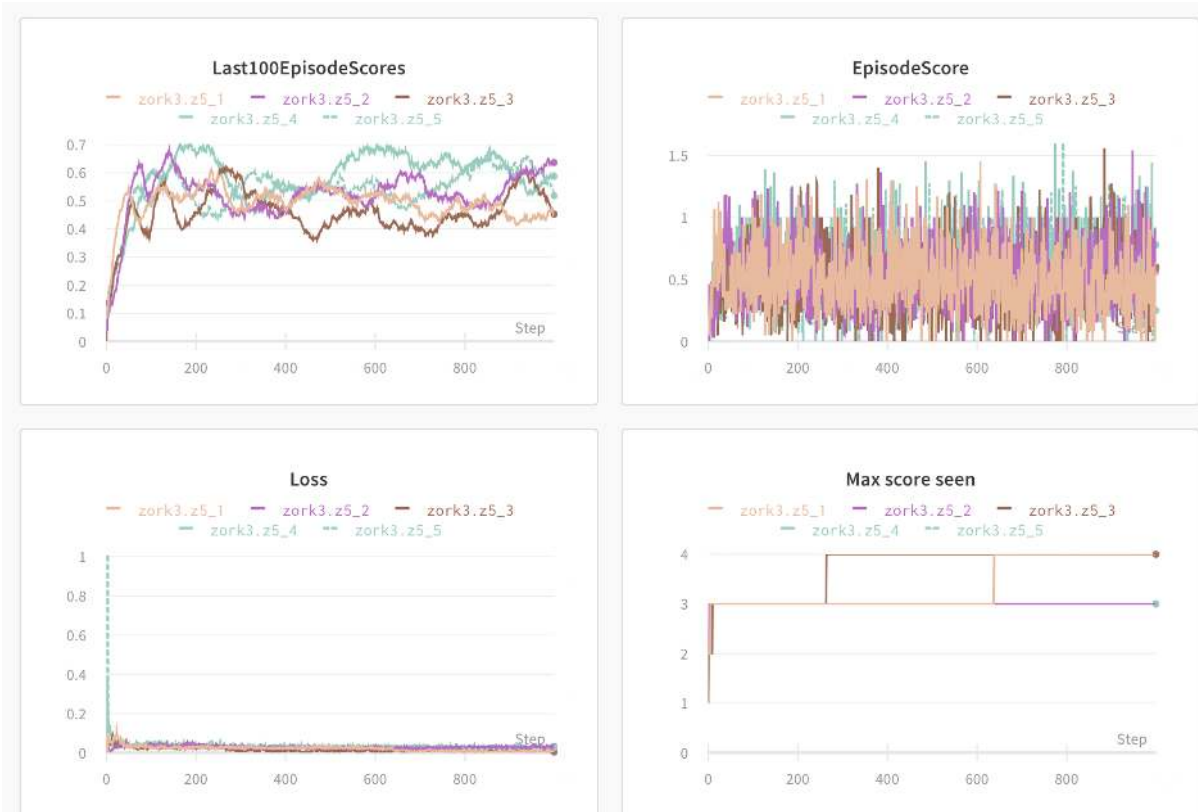


Figure 14: CALM (GPT-2) learning curves on Zork3.

State 62235: [CLS] North of House You are facing the north side of a white house. There is no door here, and all the windows are boarded up. To the north a narrow path winds through the trees. [SEP] north [SEP] Forest Path This is a path winding through a dimly lit forest. The path heads north south here. One particularly large tree with some low branches stands at the edge of the path. You are empty handed. Forest Path This is a path winding through a dimly lit forest. The path heads north south here. One particularly large tree with some low branches stands at the edge of the path. [SEP]

Actions62235: ['climb tree', 'up', 's', 'n', 'north', 'south', 'east', 'west']
 Qvalues62235: [15.38, 15.29, 12.4, 12.34, 11.99, 11.73, 11.13, 10.57]
 >>> Action62235: up
 Reward62235: 0, Score 0, Done False

State 62236: [CLS] Forest Path This is a path winding through a dimly lit forest. The path heads north south here. One particularly large tree with some low branches stands at the edge of the path. [SEP] up [SEP] Up a Tree You are about 10 feet above the ground nestled among some large branches. The nearest branch above you is above your reach. Beside you on the branch is a small bird's nest. In the bird's nest is a large egg encrusted with precious jewels, apparently scavenged by a childless songbird. The egg is covered with fine gold inlay, and ornamented in lapis lazuli and mother of pearl. Unlike most eggs, this one is hinged and closed with a delicate looking clasp. The egg appears extremely fragile. You are empty handed. Up a Tree You are about 10 feet above the ground nestled among some large branches. The nearest branch above you is above your reach. Beside you on the branch is a small bird's nest. In the bird's nest is a large egg encrusted with precious jewels, apparently scavenged by a childless songbird. The egg is covered with fine gold inlay, and ornamented in lapis lazuli and mother of pearl. Unlike most eggs, this one is hinged and closed with a delicate looking clasp. The egg appears extremely fragile. [SEP]

Actions62236: ['get egg', 'take egg', 'eat egg', 'down', 'take all', 'get nest', 'take nest']
 Qvalues62236: [17.03, 17.02, 16.99, 13.22, 11.75, 11.05, 10.68]
 >>> Action62236: eat egg
 Reward62236: 5, Score 5, Done False

State 62237: [CLS] Up a Tree You are about 10 feet above the ground nestled among some large branches. The nearest branch above you is above your reach. Beside you on the branch is a small bird's nest. In the bird's nest is a large egg encrusted with precious jewels, apparently scavenged by a childless songbird. The egg is covered with fine gold inlay, and ornamented in lapis lazuli and mother of pearl. Unlike most eggs, this one is hinged and closed with a delicate looking clasp. The egg appears extremely fragile. [SEP] eat egg [SEP] (Taken) I don't think that the jewel encrusted egg would agree with you. You are carrying: A jewel encrusted egg Up a Tree You are about 10 feet above the ground nestled among some large branches. The nearest branch above you is above your reach. Beside you on the branch is a small bird's nest. [SEP]

Actions62237: ['down', 'jump', 'put egg in nest', 'drop egg', 'throw egg at nest', 'eat nest', 'get nest', 'take nest']
 Qvalues62237: [12.93, 12.93, 11.49, 11.22, 11.1, 9.49, 9.41, 9.36]
 >>> Action62237: down
 Reward62237: 0, Score 5, Done False

State 62238: [CLS] (Taken) I don't think that the jewel encrusted egg would agree with you. [SEP] down [SEP] Forest Path You are carrying: A jewel encrusted egg Forest Path This is a path winding through a dimly lit forest. The path heads north south here. One particularly large tree with some low branches stands at the edge of the path. [SEP]

Actions62238: ['south', 'climb tree', 'drop egg', 'drop all', 'up', 'throw egg at tree', 'east', 'north', 'west']
 Qvalues62238: [14.26, 11.78, 11.46, 11.41, 11.03, 11.03, 10.85, 10.72, 10.5]
 >>> Action62238: south
 Reward62238: 0, Score 5, Done False

State 62239: [CLS] Forest Path [SEP] south [SEP] North of House You are carrying: A jewel encrusted egg North of House You are facing the north side of a white house. There is no door here, and all the windows are boarded up. To the north a narrow path winds through the trees. [SEP]

Actions62239: ['east', 'southeast', 'west', 'north', 'southwest', 'drop all', 'drop egg']
 Qvalues62239: [15.96, 15.82, 12.94, 12.86, 12.68, 12.32, 12.26]
 >>> Action62239: southeast
 Reward62239: 0, Score 5, Done False

State 62240: [CLS] North of House [SEP] southeast [SEP] 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. You are carrying: A jewel encrusted egg 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. [SEP]

Actions62240: ['drop egg', 'drop all', 'northwest', 'north', 'open window', 'south', 'southwest', 'east']
 Qvalues62240: [17.9, 17.89, 14.71, 14.5, 13.59, 13.51, 12.97, 12.8]
 >>> Action62240: drop all
 Reward62240: 0, Score 5, Done False

State 62241: [CLS] 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. [SEP] drop all [SEP] jewel encrusted egg: Dropped. You are empty handed. 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. There is a jewel encrusted egg here. [SEP]

Actions62241: ['open window', 'take it', 'get egg', 'get encrusted egg', 'eat egg', 'take egg', 'get it', 'take all', 'get all', 'north', 'northwest', 'south', 'east', 'southwest']
 Qvalues62241: [19.91, 16.52, 16.44, 16.4, 16.26, 16.25, 14.9, 14.25, 14.03, 13.86, 13.17, 12.49, 12.45, 12.1]
 >>> Action62241: open window
 Reward62241: 0, Score 5, Done False

State 62242: [CLS] jewel encrusted egg: Dropped. [SEP] open window [SEP] With great effort, you open the window far enough to allow entry. You are empty handed. 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. There is a jewel encrusted egg here. [SEP]

Actions62242: ['west', 'in', 'enter window', 'north', 'northwest', 'east', 'eat egg', 'take egg', 'get egg', 'down', 'get all', 'south', 'southwest', 'up']
 Qvalues62242: [22.42, 20.53, 18.46, 16.62, 15.93, 15.17, 14.3, 14.13, 14.1, 13.82, 13.74, 13.68, 13.56, 11.59]
 >>> Action62242: west
 Reward62242: 10, Score 15, Done False

State 62243: [CLS] With great effort, you open the window far enough to allow entry. [SEP] west [SEP] 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 You are empty handed. 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 [SEP]

Actions62243: ['open sack', 'eat sack', 'open bottle', 'take sack', 'get sack', 'out', 'enter window', 'east', 'west', 'get bottle', 'take bottle', 'take all', 'get all', 'up']
 Qvalues62243: [13.74, 13.68, 12.38, 11.53, 11.4, 11.25, 11.13, 11.06, 10.36, 10.23, 10.15, 9.63, 9.61, 6.54]
 >>> Action62243: open sack
 Reward62243: 0, Score 15, Done False

State 62244: [CLS] 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 [SEP] open sack [SEP] Opening the brown sack reveals a lunch, and a clove of garlic. You are empty handed. 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. A bottle is sitting on the table. The glass bottle contains: A quantity of water There is a brown sack here. The brown sack contains: A lunch A clove of garlic [SEP]

Actions62244: ['take sack', 'get sack', 'eat sack', 'take bag', 'take garlic', 'out', 'close sack', 'get garlic', 'take lunch', 'take clove', 'eat garlic', 'east', 'west', 'take bottle', 'get all', 'take all', 'up']

Qvalues62244: [15.32, 15.25, 15.23, 14.95, 12.16, 12.12, 11.9, 11.89, 11.84, 11.7, 11.66, 11.65, 11.18, 10.95, 10.44, 10.39, 9.46]

>> Action62244: get sack

Reward62244: 0, Score 15, Done False

State 62245: [CLS] Opening the brown sack reveals a lunch, and a clove of garlic. [SEP] get sack [SEP] Taken. You are carrying: A brown sack The brown sack contains: A lunch A clove of garlic 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. A bottle is sitting on the table. The glass bottle contains: A quantity of water [SEP]

Actions62245: ['get clove', 'take clove', 'get garlic', 'take garlic', 'west', 'put garlic on table', 'drop garlic', 'open bottle', 'put sack on table', 'get food', 'drop all', 'drop sack', 'out', 'east', 'get bottles', 'take bottle', 'get bottle', 'take glass', 'get all', 'up']

Qvalues62245: [16.25, 16.19, 16.01, 15.94, 14.53, 14.18, 14.04, 13.83, 12.39, 11.77, 11.62, 11.56, 11.05, 10.86, 10.26, 10.21, 10.08, 9.4, 9.3, 7.84]

>> Action62245: take clove

Reward62245: 0, Score 15, Done False

State 62246: [CLS] Taken. [SEP] take clove [SEP] Taken. You are carrying: A clove of garlic A brown sack The brown sack contains: A lunch 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. A bottle is sitting on the table. The glass bottle contains: A quantity of water [SEP]

Actions62246: ['west', 'open bottle', 'drop clove', 'put garlic on table', 'drop garlic', 'put clove on table', 'put clove in sack', 'put garlic in sack', 'east', 'out', 'take bottle', 'get bottle', 'get all', 'up', 'get glass', 'take glass']

Qvalues62246: [17.59, 16.09, 14.8, 14.69, 14.66, 14.47, 13.37, 13.26, 12.45, 12.12, 11.82, 11.51, 10.67, 10.48, 10.48, 10.38]

>> Action62246: west

Reward62246: 0, Score 15, Done False

State 62247: [CLS] Taken. [SEP] west [SEP] Living Room You are in the living room. There is a doorway to the east, a wooden door with strange gothic lettering to the west, which appears to be nailed shut, a trophy case, and a large oriental rug in the center of the room. Above the trophy case hangs an elvish sword of great antiquity. A battery powered brass lantern is on the trophy case. You are carrying: A clove of garlic A brown sack The brown sack contains: A lunch Living Room You are in the living room. There is a doorway to the east, a wooden door with strange gothic lettering to the west, which appears to be nailed shut, a trophy case, and a large oriental rug in the center of the room. Above the trophy case hangs an elvish sword of great antiquity. A battery powered brass lantern is on the trophy case. [SEP]

Actions62247: ['light lantern', 'get lantern', 'take lantern', 'turn off lantern', 'open case', 'open trophy case', 'turn on lantern', 'east', 'take all', 'get sword', 'get all', 'take sword', 'drop all']

Qvalues62247: [20.04, 19.1, 18.96, 18.95, 18.35, 18.35, 17.76, 15.08, 14.41, 14.23, 14.12, 13.97, 12.25]

>> Action62247: light lantern

Reward62247: 0, Score 15, Done False

State 62248: [CLS] Living Room You are in the living room. There is a doorway to the east, a wooden door with strange gothic lettering to the west, which appears to be nailed shut, a trophy case, and a large oriental rug in the center of the room. Above the trophy case hangs an elvish sword of great antiquity. A battery powered brass lantern is on the trophy case. [SEP] light lantern [SEP] (Taken) The brass lantern is now on. You are carrying: A brass lantern (providing light) A clove of garlic A brown sack The brown sack contains: A lunch Living Room You are in the living room. There is a doorway to the east, a wooden door with strange gothic lettering to the west, which appears to be nailed shut, a trophy case, and a large oriental rug in the center of the room. Above the trophy case hangs an elvish sword of great antiquity. [SEP]

Actions62248: ['take sword', 'get sword', 'turn off lantern', 'drop sack', 'drop all', 'drop lantern', 'east']

Qvalues62248: [22.64, 22.46, 19.89, 17.26, 15.66, 15.57, 14.99]

>> Action62248: get sword

Reward62248: 0, Score 15, Done False

State 62249: [CLS] (Taken) The brass lantern is now on. [SEP] get sword [SEP] Taken. You are carrying: A sword A brass lantern (providing light) A clove of garlic A brown sack The brown sack contains: A lunch Living Room You are in the living room. There is a doorway to the east, a wooden door with strange gothic lettering to the west, which appears to be nailed shut, a trophy case, and a large oriental rug in the center of the room. [SEP]

Actions62249: ['move rug', 'drop sword', 'turn off lantern', 'east', 'put lantern in case', 'up']

Qvalues62249: [25.33, 15.03, 13.93, 11.99, 11.17, 10.45]

>> Action62249: move rug

Reward62249: 0, Score 15, Done False

State 62250: [CLS] Taken. [SEP] move rug [SEP] With a great effort, the rug is moved to one side of the room, revealing the dusty cover of a closed trap door. You are carrying: A sword A brass lantern (providing light) A clove of garlic A brown sack The brown sack contains: A lunch Living Room You are in the living room. There is a doorway to the east, a wooden door with strange gothic lettering to the west, which appears to be nailed shut, a trophy case, and a closed trap door at your feet. [SEP]

Actions62250: ['open trap door', 'open trapdoor', 'open cover', 'drop lantern', 'east']

Qvalues62250: [27.97, 27.94, 27.91, 21.05, 20.41]

>> Action62250: open cover

Reward62250: 0, Score 15, Done False

State 62251: [CLS] With a great effort, the rug is moved to one side of the room, revealing the dusty cover of a closed trap door. [SEP] open cover [SEP] The door reluctantly opens to reveal a rickety staircase descending into darkness. You are carrying: A sword A brass lantern (providing light) A clove of garlic A brown sack The brown sack contains: A lunch Living Room You are in the living room. There is a doorway to the east, a wooden door with strange gothic lettering to the west, which appears to be nailed shut, a trophy case, and a rug lying beside an open trap door. [SEP]

Actions62251: ['down', 'close trap door', 'east', 'drop sack', 'drop lantern', 'put sack in case']

Qvalues62251: [30.88, 25.45, 22.21, 21.96, 20.7, 20.62]

>> Action62251: down

Reward62251: 25, Score 40, Done False

Figure 15: Last episode of game trajectory of DRRN + CALM (GPT-2) on Zork 1, from start until 40 is scored.

Example 1: "You see the monster stumble from its cave. You are carrying a bow and three arrows"

CALM (GPT-2) Top 10 Generated Actions: ['south',
'hit monster with bow',
'up',
'shoot monster with bow',
'down',
'east',
'west',
'north',
'kill monster',
'shoot monster']

Example 2: "Tom looked concerned. The panel of levers and dials clearly was confusing him"

CALM (GPT-2) Top 10 Generated Actions: ['south',
'pull lever',
'talk to tom',
'open panel',
'east',
'west',
'turn dials',
'north',
'push button',
'pull levers']

Example 3: "Your body feels cold as you plunge into the river"

CALM (GPT-2) Top 10 Generated Actions: ['south',
'wait',
'up',
'enter river',
'down',
'east',
'west',
'swim',
'drink water',
'north']

Figure 16: Some handpicked example observations and top 10 action predictions for CALM (GPT-2). The top non-directional actions demonstrate some understanding of the objects present in the observations, and some commonsense actions involving those objects.