GOAL RANDOMIZATION FOR PLAYING TEXT-BASED GAMES WITHOUT A REWARD FUNCTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Playing text-based games requires language understanding and sequential decision making. The objective of a reinforcement learning agent is to behave so as to maximise the sum of a suitable scalar reward function. In contrast to current RL methods, humans are able to learn new skills with little or no reward by using various forms of intrinsic motivation. We propose a goal randomization method that leverages human intelligence to use random basic goals to train a policy in the absence of the reward of environments. Specifically, through simple but effective goal generation, our method learns to continuously propose challenging – yet temporal and achievable – goals that allow the agent to learn general skills for acting in a new environment, independent of the task to be solved. In a variety of text-based games, we show that this simple method results in competitive performance for agents. We also show that our method can learn policies that generalize across different text-based games. In further, we demonstrate an interesting result that our method works better than one of state-of-the-art agents GATA, which uses environment rewards for some text-based games.

1 INTRODUCTION

Text-based games are complex, interactive simulations in which the game state is described with text and players act using simple text commands (e.g., take sandwich from table, eat sandwich, open door, etc.) (Côté et al., 2018). They serve as a proxy for studying how agents can exploit language to comprehend and interact with the environment. Text-based games are a useful challenge in the pursuit of intelligent agents that communicate with humans (e.g., in customer service systems). Inspired by this, one of the long term goals in AI is to build agents that can learn to accomplish tasks with language. In the domain of text-based games, the key challenge is to decipher the long textual observations, extract reward cues from them, and generate semantically rich representations such that the policy learned on top of it is well informed. Most of the existing works learn to model text representations during the RL training (Hausknecht et al., 2020; Ammanabrolu & Hausknecht, 2020). Some works also study generalizability of games with different difficulty levels or layouts (Ammanabrolu & Riedl, 2019a; Adolphs & Hofmann, 2020). Deep reinforcement learning (RL) has been demonstrated to effectively learn to solve reward-driven problems in various tasks (Mnih et al., 2013; Silver et al., 2016; Schulman et al., 2017). In contrast, humans are able to learn new skills with little or no reward by using various forms of intrinsic motivation.

Playing text-based games without a reward function is an exceedingly challenging problem. We consider the setting where reward functions are unknown, so we want to learn an agent that can drive itself without environmental rewards. Learning agents without reward has several practical applications (Eysenbach et al., 2018). Environments with sparse rewards effectively have no reward until the agent randomly reaches a goal state. Learning intelligent agents without supervision may help address challenges in exploration in these environments (Gupta et al., 2018; Sharma et al., 2019). In many practical settings, interacting with the environment is essentially free, but evaluating the reward requires human feedback (Christiano et al., 2017). However, there is no work on playing text-based games without reward functions. Solving such kinds of non-reward tasks can encourage agents to explore, experiment, and invent. Sometimes, as in many games and fantasies, without any direct link to reality or to any source of extrinsic reward, it is crucial to enable learning in real-world environments, even for humans (Schulz, 2012).

In this paper, we take a step towards agents that can learn from text-based games without reward functions. The environments are designed to mimic some real-world scenarios where there are no reward cues for guiding agents. The goal is for an agent to be able to learn one policy that is able to solve both tasks it was trained on as well as a variety of unseen tasks which contain similar tasks as the training tasks. To do this, we propose a new method for learning an agent with deep RL in the absence of any rewards. We use a set of available goals characterized by natural language via common-sense rules, and select one of them according to the current knowledge graph based observation. Then, we learn policies for goal-conditioned reinforcement learning. Specifically, our method works as follows. Whenever an agent builds a knowledge graph of the textural world, our method gives a goal in natural language to the agent based on the knowledge graph. The agent takes the goal to form a new experience with a corresponding intrinsic reward, alleviating the no reward problem. For example, our method can describe what the agent has achieved in the episode, and the agent can use goals as advice to obtain intrinsic rewards. In addition, our method also provides a time limit for a goal. If the agent can not accomplish the goal in the time limit, it can use a new goal replacing the old one. The agent can have the opportunity to get out of the difficult goals. We show many benefits brought by language goal representation when combined with goal advice. The agent can efficiently solve reinforcement learning problems in challenging text-based environments; it can generalize to unseen instructions, and even generalize to instruction with unseen lexicons.

Our method can be viewed as the intrinsic motivation of any agent trained with policy gradient-based methods (Oudeyer & Kaplan, 2009; Barto, 2013). Most intrinsic methods design intrinsic motivations to assist environmental rewards to make agents learn efficiently. However, our method is to use the intrinsic motivation to play text-based games without environmental rewards. Under this view, we also need to encode the intrinsic motivation into the policy. That is, the original policy network becomes a goal-conditional policy; the goal advice can then be seen as a “bolt-on” to the original policy network. Because we use natural language for self-advice. It is flexible and can be used on a variety of RL training model architectures and training settings by encoding the intrinsic motivation.

In summary, we make the following contributions: (i) we first study the problem of playing text-based games without any reward functions and propose a new goal randomization method for solving the problem;[1] (ii) we show, through common-sense knowledge, that agents trained with goal randomization gradually learn to interact with the environment and solve tasks which are difficult for state-of-the-art methods; (iii) we perform an extensive qualitative analysis and ablation study, and we also find an interesting result that our method works better than one of state-of-the-art agents GATA (Adhikari et al., 2020), which uses environment rewards, for some text-based games.

2 RELATED WORK

Reinforcement learning for text-based games. Existing agents either perform based on predefined rules or learn to make responses by interacting with the environment. Rule-based agents (Atkinson et al., 2019; Fulda et al., 2017; Hausknecht et al., 2019; Kostka et al., 2017) attempt to solve text-based games by injecting heuristics. They are thus not flexible since a huge amount of prior knowledge is required to design rules (Hausknecht et al., 2020). Learning-based agents (Adolphs & Hofmann, 2020; Hausknecht et al., 2020; He et al., 2016; Jain et al., 2020; Narasimhan et al., 2015; Yin & May, 2019; Yuan et al., 2018; Zahavy et al., 2018) usually employ deep reinforcement learning algorithms to deliver adaptive game solving strategies. KG-based agents have been developed to enhance the performance of learning-based agents with the assistance of KGs. KGs can be constructed by simple rules so that it substantially reduces the amount of prior knowledge required by rule-based agents. While KGs have been leveraged to handle partial observability (Ammanabrolu & Hausknecht, 2020; Ammanabrolu & Riedl, 2019a; Zelinka et al., 2019), reduce action space (Ammanabrolu & Hausknecht, 2020; Ammanabrolu & Riedl, 2019a), and improve generalizability (Adhikari et al., 2020; Ammanabrolu & Riedl, 2019b). Recently, Murugesan et al. (2020) tried to introduce commonsense reasoning for playing synthetic games. While these works all follow the standard setting with reward functions, our work is the first work that trains an agent without reward functions.

[1] Code can be found here: `https://anonymous.4open.science/r/goalrand-E167/`

Intrinsic motivation for reinforcement learning. Intrinsic motivation has been widely used in reinforcement learning (Oudeyer et al., 2007; Oudeyer & Kaplan, 2009; Barto, 2013). It has been proven effective for solving various hard-exploration tasks (Bellemare et al., 2016; Pathak et al., 2017; Burda et al., 2018b). One prominent formulation is the use of novelty, which in its simplest form can be estimated with state visitation counts (Strehl & Littman, 2008) and has been extended to high-dimensional state spaces (Bellemare et al., 2016; Burda et al., 2018b; Ostrovski et al., 2017). Other sophisticated versions of curiosity (Schmidhuber, 1991) guide the agent to learn about environment dynamics by encouraging it to take actions that reduce the agent's uncertainty (Stadie et al., 2015; Burda et al., 2018b), have unpredictable consequences (Pathak et al., 2017; Burda et al., 2018a), or a large impact on the environment (Raileanu & Rocktäschel, 2020). Other forms of intrinsic motivation include empowerment (Klyubin et al., 2005) which encourages control of the environment by the agent, and goal diversity (Pong et al., 2019) which encourages maximizing the entropy of the goal distribution. Intrinsic goals are discovered from language supervision (Lair et al., 2019). Except for exploration, intrinsic motivation is also used for other problems, such as evolutionary (Singh et al., 2009; Sorg et al., 2010; Zheng et al., 2018). Most works use intrinsic motivation as additive rewards for environmental rewards. Our work differs from those works by formulating intrinsic motivation to train an agent directly for text-based games.

Generalization in text-based games. Generalization is a challenging problem for reinforcement learning (Tobin et al., 2017; Agarwal et al., 2019). In text-based games, it is difficult to study generalization in games initially designed for human players (Hausknecht et al., 2020), as they are so challenging that existing RL agents are still far from being able to solve a large proportion of them even under the single game setting (Yao et al., 2020). Furthermore, these games usually have different themes, vocabularies and logics, making it hard to determine the domain gap (Ammanabrolu & Riedl, 2019b). Compared with these man-made games, the synthetic games (Côté et al., 2018; Urbanek et al., 2019) provide a more natural way to study generalization by generating multiple similar games with customizable domain gaps (e.g., by varying game layouts). Generally, the training and testing game sets in previous works have either the same difficulty level (Ammanabrolu & Riedl, 2019a; Murugesan et al., 2021), or a mixture of multiple levels (Adolphs & Hofmann, 2020; Yin et al., 2020), or both (Adhikari et al., 2020). Our works can be used for generalization in games. Moreover, our work does not use the reward functions of environments to train an agent.

3 Preliminaries

Text-based games as POMDPs. Text-based games can be formally described as Partially Observable Markov Decision Processes (POMDPs). POMDPs can be defined as a tuple $\langle \boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{A}}, \boldsymbol{T}, \boldsymbol{R}, \boldsymbol{\Omega}, \boldsymbol{O}, \boldsymbol{\gamma} \rangle$: the state set $\boldsymbol{\mathcal{S}}$, the action set $\boldsymbol{\mathcal{A}}$, the state transition probabilities $\boldsymbol{T}$, the reward function $\boldsymbol{R}$, the observation set $\boldsymbol{\Omega}$, the conditional observation probabilities $\boldsymbol{O}$ and the discount factor $\boldsymbol{\gamma} \in (0, 1]$. At each time step, the agent will receive a textual observation $\boldsymbol{o}_t \in \boldsymbol{\Omega}$, depending on the current state and previous action via the conditional observation probability $O(\boldsymbol{o}_t|\boldsymbol{s}_t, \boldsymbol{a}_{t-1})$. By executing an action $\boldsymbol{a}_t \in \boldsymbol{\mathcal{A}}$, the environment will transit into a new state based on the state transition probability $T(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$, and the agent will receive the reward $\boldsymbol{r}_{t+1} = R(\boldsymbol{s}_t, \boldsymbol{a}_t)$. Same as Markov Decision Process (MDPs), the goal of the agent is to learn an optimal policy $\boldsymbol{\pi}^*$ to maximize the expected future discounted sum of rewards from each time step: $\boldsymbol{R}_t = \mathbb{E}[\sum_{k=0}^{\infty} \boldsymbol{\gamma}^k \boldsymbol{r}_{t+k+1}]$.

Knowledge graph for text-based games. Graph-based representations are effective for text-based games because the state in these games adheres to a graph-like structure. The content in most observations of the environment corresponds either to entity attributes or to relational information about entities in the environment. Knowledge graph (KG) for a text-based game can be built from a set of triplets $\langle Subject,\ Relation,\ Object \rangle$, denoting that the $Subject$ has $Relation$ with the $Object$. For example, $\langle Kitchen,\ Has,\ Food \rangle$. The KG is denoted as $G = (V, E)$, where V and E are the node set and the edge set, respectively. Both $Subject$ and $Object$ belong to the node set V. $Relation$, which corresponds to the edge connecting them, belongs to E.

4 Methodology

4.1 Problem statement

We aim to design an RL-based agent that is able to solve text-based games without reward functions. We focus on man-made text-based games, which are initially designed for human players (Hausknecht et al., 2020). These games are devised with more complex logic and much larger action space than synthetic games (Côté et al., 2018). Most of them share similar themes and vocabularies. There are no obvious goals for accomplishing games. There are also no rewards provided by the text-based games or environments. At every step we construct an input $\boldsymbol{s}_t$ as the combination of three components: a textual observation $\boldsymbol{o}_{t,\text{text}}$ and a KG $\boldsymbol{o}_{t,\text{KG}}$ (note that here $\boldsymbol{s}_t$ should not be regarded as a true game state as the games are not fully observable). $\boldsymbol{o}_{t,\text{text}}$ further includes the current state $\boldsymbol{o}_{t,\text{desc}}$ (describing the environment), inventory $\boldsymbol{o}_{t,\text{inv}}$ (describing items collected by a player), game feedback $\boldsymbol{o}_{t,\text{feed}}$, and previous action taken $\boldsymbol{a}_{t-1}$. While $\boldsymbol{o}_{t,\text{text}}$ mainly reflects the current observation, $\boldsymbol{o}_{t,\text{KG}}$ records the game history. Therefore, the KG can help the agent to handle partial observability. At each time step, the triples extracted from the current textual observation $\boldsymbol{o}_{t,\text{text}}$ are used to update the KG as $\boldsymbol{o}_{t,\text{KG}} = \text{GraphUpdate}(\boldsymbol{o}_{t-1,\text{KG}}, \boldsymbol{o}_{t,\text{text}})$.

We consider two scenarios for text-based games: seen levels, where the training and testing games have the same levels, but different layouts, and unseen levels, where the training and testing games are different in levels and layouts. Many games share similar themes and vocabularies, but vary in their layouts and / or difficulty levels. For example, games of the cooking theme (Côté et al., 2018) share the same overall objective: prepare the meal. The layout of a game contains the room connectivity and the preparing steps (e.g., the type / location of ingredients). The difficulty of a game depends on the complexity of the map (e.g., the number of rooms) and the recipe (e.g., the number of ingredients), such that two games with different levels are naturally different in their layouts. More examples are provided in Appendix A.

4.2 Goal randomization

Text-based games use texts for describing what the player or agent needs to do to win the game. The goal of the agent in the game is to complete some objective. Such textual information hints about the objective/purpose of the game, what dangers are present, and provides clues that might become handy later in the game for solving puzzles. However, without rewards, it becomes difficult for the agent to learn to win a game and generalize to other environments. The agent must create intrinsic motivation by itself to learn new skills in the textual world.

The purpose of goal randomization is to provide enough motivation variability at training time such that at test time the agent is able to generalize to other environments. Goal randomization provides a way to guide the agent to learn essential skills. With goal randomization, the agent can obtain experiences and skills by accomplishing tasks with random goals. While a whole game may be difficult to accomplish due to long-term temporal dependency, decomposing it into a sketch of essential skills or subtasks will make the game easier to be solved (Sohn et al., 2018; Shiarlis et al., 2018). If we consider the solving strategy for a subtask as a skill, the generalizability for an unseen game will also be improved by recomposing the learnt skills. We can characterize a subtask by a very basic goal. In text-based games, we make the goal to be instruction-like textual descriptions (e.g., "find purple potato"), yielding better flexibility and interpretability than using a state as the goal (Schaul et al., 2015; Andrychowicz et al., 2017; Plappert et al., 2018).

Figure 1 shows the overview of our method, which consists of a goal set generator and a goal-conditional policy. We denote the set containing all required goals for solving a game as $\mathcal{G}$. Inspired by (Jiang et al., 2019), the goal set generator can be implemented by different approaches, including supervised language models and non-learning-based methods such as human supervisors and functional programs. In our work, simply we use common-sense rules to obtain $\mathcal{G}_t$, defined as follows:

- (I) if an ingredient i is not collected in the knowledge graph, then the goal is "find i";
- (II) if an ingredient i is collected and has a requirement q in the knowledge graph, then the goal is "q i".

In the real world, the goal of some task can be complex and depend on other goals. However, these generated goals are simple and basic for text-based games. On one hand, these goals are common and shareable for different tasks and environments. On the other hand, these goals can be easily described using a simple phrase or sentence. For example, "find an object", "prepare something" or "eat something". The agent can learn some essential skills for finishing these tasks with basic and simple goals. Common-sense rules or human instruction can provide a lot of such basic and simple

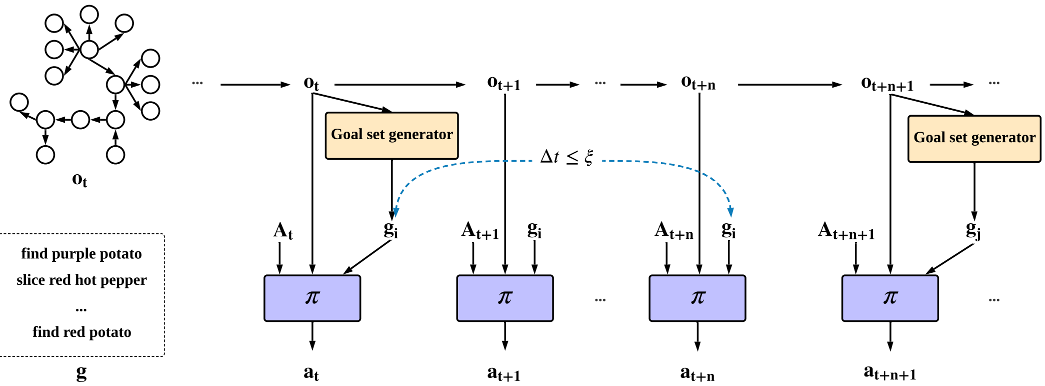


Figure 1: The overview of our method. The observation $\boldsymbol{o}_t$ is represented as a knowledge graph. $\boldsymbol{g}$ shows an example of the set of available goals. At each time step t, the policy π receives the observation $\boldsymbol{o}_t$, a goal $\boldsymbol{g}_i$, and selects an action $\boldsymbol{a}_t$ from the set of action candidates $\boldsymbol{A}_t$. We also consider the goal time limit Δt, that the agent will be forced to re-sample a new goal if it fails to accomplish the goal in ξ steps.

fundamental goals and make goals human-interpretable and consistent with existing games: e.g., food items can be combined, cooked, and eaten. More details are in Appendix B.

We train the agent by generating random goals during games. We randomize the following aspects of the goals for the agent used during training: different types of goals and different objects. Language can provide an abstract representation for a goal. This motivates our proposal to use natural language for representing the goal space. Since we use language to represent goals, we can use natural language to represent different goals easily. Concretely, we can define a lot of goals according to common-sense rules and knowledge. They are natural language descriptions of the goals. Then we have a goal set $\mathcal{G}$. With the goal set, we introduce a new objective for the agents:

$$\theta^* = \arg\min_{\theta} \mathcal{L}(\pi_\theta; g), g \sim \mathcal{G}. \tag{1}$$

We assume that the goal is also time-awareness. A goal is bounded by a time limit $[0, \xi]$. Thus there is a time limit to reach a goal. If a goal is not finished in its time limit, we sample a new goal to replace the old goal. It means during an episode, there may be more than one goal. One desirable property is goal diversity (Pong et al., 2019; Raileanu & Rocktäschel, 2020; Campero et al., 2020). In our implementation, we use uniform sampling that encourages such diversity. This sampling strategy, along with the time limits of the goals, helps the agent avoid getting stuck in local minima.

Without any reward functions of environments, we design the following pseudo-reward:

$$r(s_t, a_t; g_i) = f_{g_i}(s_{t+1}), \text{where } f_{g_i} : S \to \{0, 1\}. \tag{2}$$

The pseudo-reward is general for different tasks and environments because it just involves essential goals. During goal randomization, we sequentially sample goals for each episode. Each goal is delivered as a subtask for the agent to act throughout the episode. In each subtask that is time-limited, the learning objective is to optimize the expectation of the return $G_{t:t+\Delta t}$.

4.3 Model Architecture and Training

We use the goal-oriented reinforcement learning framework following (Schaul et al., 2015). We augment the previously defined infinite-horizon, discounted POMDPs with a goal space $\mathcal{G}$. A goal g is chosen from $\mathcal{G}$. The difference of our model is that g varies for each episode. The goal induces a reward function $r_g : S \times A \to R$, that assigns reward to a state conditioned on a given goal. The policy π follows the goal-conditioned RL setting (Kaelbling, 1993), where a_t is selected by considering both o_t^{KG} and g. Figure 1 shows the architecture of π, which is constructed based on both o_t^{KG} and g. π has the graph encoder and text encoder to process game information.

Algorithm 1 Goal Randomization (GoalRand)

Require: RL algorithm π, goal set $\mathcal{G}$, replay buffer $\mathcal{B}$
Initialize π
for episode = $1, 2, \cdots, M$ **do**
  Sample an initial goal g from $\mathcal{G}$
  Set the time limit of g: $\Delta t = 0$
  for $t = 0, \cdots, T-1$ **do**
    if $\Delta t = \xi$ or g is done **then**
      Sample a new goal g from $\mathcal{G}$
      Set the time limit of g: $\Delta t = 0$
    end if
    Sample action $a_t \sim \pi$
    Execute the action a_t and observe a new state s_{t+1}
    Set the pseudo-reward reward r_t according to (2)
    Store the translation to $\mathcal{B}$
    $\Delta t = \Delta t + 1$
  end for
  for $t = 1, \cdots, N$ **do**
    Sample a minibatch B from the replay buffer $\mathcal{B}$
    Optimize π using the minibatch B
  end for
end for

To integrate language representation of goals into the model, we design the architecture as follows. The architecture consists of three modules. Similar to previous works (Adhikari et al., 2020), we use a graph encoder to encode o_t^{KG} as state representation $\boldsymbol{s}_t$, and a text encoder to encode $\mathcal{G}_t$ as a stack of goal representations. Arbitrary graph encoders and text encoders can be applied. We use the graph encoder based on the Relational Graph Convolutional Networks (R-GCNs) (Schlichtkrull et al., 2018) to take both nodes and edges into consideration. For the text encoder, a simple single-block transformer (Vaswani et al., 2017) is sufficient as the goal candidates are short texts. As this work does not aim at handling the combinatorial action space, we consider the admissible action set $A_t \subseteq \mathcal{A}$ for each time step (He et al., 2016). We denote an action as "admissible" if it does not lead to meaningless feedback (e.g., "Nothing happens"). The action scorer will pair $\boldsymbol{s}_t$ with each candidate $a_i \in A_t$, followed by linear layers to compute the action scores.

We name our method as GoalRand and detailed procedures are given in Algorithm 1. The goal set updates during an episode. Experiences (or transitions) from the agent are collected at every time step. In an episode, each goal g is randomly drawn and updated if the time limit of the goal terminates. For the agent, at every time step, an action is drawn based on π with a goal. The model parameters θ are periodically updated by drawing experiences from replay memories.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

We conduct experiments on multiple levels of cooking games (Côté et al., 2018). We consider two scenarios for the experiments: (1) seen levels, where the training and testing games have the same levels, but different layouts. (2) unseen levels, where the training and testing games are different in levels and layouts. While previous work (Adhikari et al., 2020) considered either a single level, or a mixture of 4 levels, we extend their setting to 8 levels. Based on the rl.0.1 game set[2], we build a training game set $\mathcal{D}_{\text{train}}$ with 4 levels, including 100 games per level. We build a validating game set $\mathcal{D}_{\text{val}}$ with the same 4 levels of $\mathcal{D}_{\text{train}}$, where each level contains 20 games. We build two testing game sets: $\mathcal{D}_{\text{test}}^{\text{seen}}$, and $\mathcal{D}_{\text{test}}^{\text{unseen}}$, both of which contain 4 levels and 20 games per level. The levels within $\mathcal{D}_{\text{test}}^{\text{seen}}$ have been seen in $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{val}}$, while there is no overlapping game. The levels within $\mathcal{D}_{\text{test}}^{\text{unseen}}$ are unseen during training. Table 1 shows the game statistics averaged over each level, where "S#" denotes a seen level and "US#" denotes an unseen level. We set the step limit of an episode as 50 for

[2] https://aka.ms/twkg/rl.0.1.zip

Table 1: Game statistics. “#Ings” denotes the number of ingredients, “#Reqs” denotes the requirements, and “#Acts” denotes the admissible actions per time step.

Level	#Triplets	#Rooms	#Objs	#Ings	#Reqs	#Acts	MaxScore
S1	21.44	1	17.09	1	1	11.54	4
S2	21.50	1	17.49	1	2	11.81	5
S3	46.09	9	34.15	1	0	7.25	3
S4	54.54	6	33.41	3	2	28.38	11
US1	19.85	1	16.01	1	0	7.98	3
US2	20.74	1	16.69	1	1	8.87	4
US3	33.04	6	24.81	1	0	7.61	3
US4	47.31	6	31.09	3	0	13.90	5

Table 2: The testing performance of models at the end of training.

Model	S1	S2	S3	S4	Avg Seen
GoalRand (ours)	**0.79**$\pm$0.12	**0.74**$\pm$0.16	**0.42**$\pm$0.06	**0.47**$\pm$0.24	**0.61**$\pm$0.14
BeBold	0.25$\pm$0.01	0.32$\pm$0.02	0.13$\pm$0.01	0.13$\pm$0.05	0.21$\pm$0.01
Random	0.27$\pm$0.02	0.27$\pm$0.01	0.25$\pm$0.02	0.19$\pm$0.02	0.24$\pm$0.01
Model	**US1**	**US2**	**US3**	**US4**	**Avg Unseen**
GoalRand (ours)	**0.94**$\pm$0.06	**0.89**$\pm$0.08	**0.72**$\pm$0.03	**0.34**$\pm$0.03	**0.72**$\pm$0.05
BeBold	0.33$\pm$0.00	0.42$\pm$0.02	0.12$\pm$0.01	0.18$\pm$0.04	0.26$\pm$0.02
Random	0.44$\pm$0.06	0.38$\pm$0.04	0.31$\pm$0.06	0.29$\pm$0.02	0.35$\pm$0.03

training and 100 for validation / testing. We train the models for 60,000 episodes. For every 1,000 episodes, we validate the model on $\mathcal{D}_{\text{val}}$, and report the testing performance on $\mathcal{D}_{\text{test}}^{\text{seen}}$ and $\mathcal{D}_{\text{test}}^{\text{unseen}}$.

5.2 Baselines

As far as we know, there is no prior work on playing text-based games without reward functions. To evaluate the performance of our algorithm, we consider constructing two baselines. One baseline is a random agent and takes actions randomly. The other baseline is a reward-free version of BeBold (Zhang et al., 2020). The intrinsic rewards are often used as additive rewards for environmental rewards. For a fair comparison, all methods do not use environmental rewards. We evaluated the following methods:

- Random: this is an agent that takes actions randomly. At each time step, the action to be executed is uniformly sampled from available.
- BeBold (Zhang et al., 2020): it is one of the state-of-the-art methods using intrinsic rewards. It provides count-based reward r_t^{count} as intrinsic reward. It counts the visitation of observations within an episode, and the accumulated visitation throughout the training process, defined as:

$$r_{t+1}^{\text{count}} = \max\left(\frac{1}{\mathrm{N}_{\text{acc}}(o_t^{\text{KG}})} - \frac{1}{\mathrm{N}_{\text{acc}}(o_{t+1}^{\text{KG}})}, 0\right) \cdot \mathbb{I}\{\mathrm{N}_{\text{epi}}(o_{t+1}^{\text{KG}}) = 1\}, \tag{3}$$

  where N_{acc} and N_{epi} denote the accumulated and episodic visitation count, respectively. The $\mathbb{I}$ operation returns 1 if o_{t+1}^{KG} is visited for the first time in the current episode, otherwise 0.
- GoalRand (ours): this is our method that performs training with random basic goals. We use $\xi = 5$ as the goal time limit. Without receiving true reward signals from environments, we generate pseudo-reward for training agents to solve different tasks.

5.3 Main results

Table 2 shows the testing performance at the end of training, and Figure 2 shows testing performance with respect to the training episodes. We use the normalized score, which is defined as the collected score normalized by the maximum available score for this game (“MaxScore” in Table 1), to measure the performance. The proposed GoalRand outperforms the baselines in both seen and unseen levels.

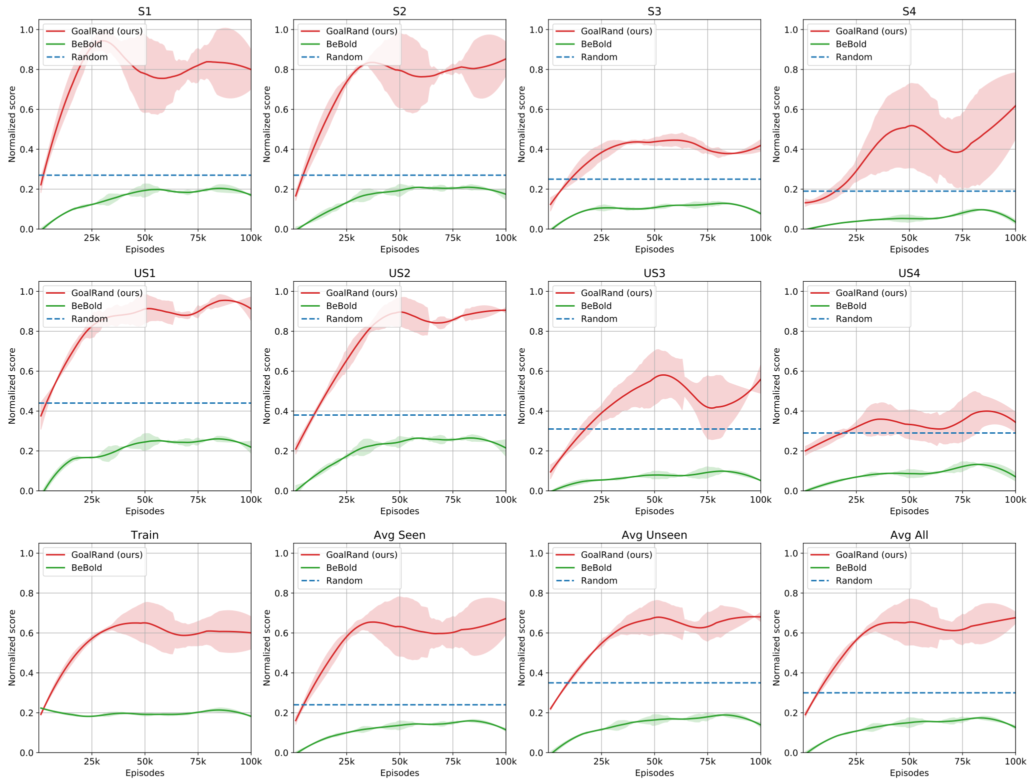


Figure 2: The testing results on the games of seen and unseen levels with respect to the training episodes. The dashed line denotes the random agent, which is non-learnable. The first row shows the results on games in the seen levels. The second row shows the results on games in the unseen levels. The third row shows the summary results for different methods. Our GoalRand significantly outperforms the BeBold method and the random baseline.

Table 3: The testing performance of models at the end of training.

Model	S1	S2	S3	S4	Avg Seen
GoalRand (ours)	**0.79**±0.12	0.74±0.16	0.42±0.06	**0.47**±0.24	**0.61**±0.14
GoalRand w/o Threshold	0.62±0.14	**0.85**±0.09	0.39±0.09	0.30±0.15	0.54±0.10
GATA	0.60±0.22	0.42±0.09	**0.48**±0.03	0.23±0.02	0.43±0.07
Model	**US1**	**US2**	**US3**	**US4**	**Avg Unseen**
GoalRand (ours)	**0.94**±0.06	**0.89**±0.08	**0.72**±0.03	0.34±0.03	**0.72**±0.05
GoalRand w/o Threshold	0.93±0.06	**0.89**±0.06	0.59±0.10	**0.38**±0.11	0.69±0.07
GATA	0.64±0.19	0.47±0.09	0.62±0.10	0.30±0.05	0.51±0.07

It solves most of the simple levels ("S1", "S2") and completes 50% of the hard levels ("S3", "S4"). In the unseen levels, where both the layouts and complexities are different from those training games, our method also achieves good performance. We also observed that only encouraging exploration is not sufficient when the environment reward is not available: the BeBold baseline learns very slow, and performs even worse than the non-learnable random agent.

5.4 Ablation Study

We then conduct ablation studies to further investigate the contributions of GoalRand's components. We consider following variants:

- GoalRand w/o Threshold: our agent without the goal time limit. That is, the agent will keep being conditioned a same goal until this goal is not available.

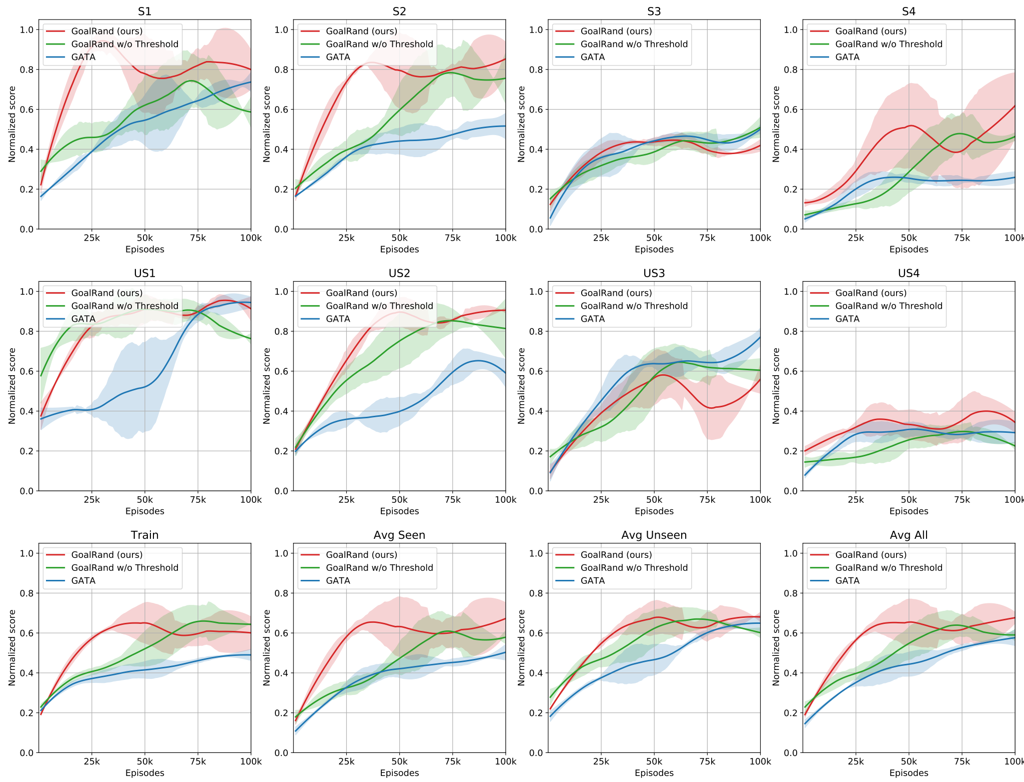


Figure 3: The ablation results on the games of seen and unseen levels with respect to the training episodes. The first row shows the results on games in the seen levels. The second row shows the results on games in the unseen levels. The third row shows the summary results for different methods. For most of the games, the results show that the goal time limit can contribute to the performance and our method works better than GATA.

- GATA (Adhikari et al., 2020): we adopt the GATA agent, which utilizes the environment reward for learning.

Table 3 shows the testing performance at the end of training, and Figure 3 shows testing performance with respect to the training episodes. The results show that the goal time limit can contribute to the performance, especially in more difficult levels such as "S4" and "US4" ("GoalRand" v.s., "GoalRand w/o Threshold"). In these levels, there are more available goals at each time step, the agent will end up failing a game if it selects a too difficult goal and gets stuck in solving it. Setting a time limit for the goal helps the agent to prevent such condition, as it can choose another goal if the current goal is too hard to solve. In the simpler levels such as "S1" and "S2", there may be fewer available goals, therefore re-sampling a new goal might be less effective. Although the GATA agent is provided with the environment reward, it shows worse performance in comparison to our agent. One possible reason is that the agent may have difficulties in solving a whole game, which is with long term dependency. Our goal randomization method helps the agent to effectively explore the environment, and simplify a complex task as a set of easier-to-solve sub-goals.

6 CONCLUSION

In this paper, we present goal randomization, a method for playing text-based games without a reward function. We show that the proposed goal randomization method via common-sense rules can learn skills for complex tasks, often solving benchmark tasks with the learned skills without actually receiving any environment rewards. The experiments demonstrate that our proposed method can improve the generalization of the agent and works well for the games of the seen and unseen levels. Interestingly, for some text-based games, the experiments show that the proposed method is better than one of state-of-the-art method GATA, which uses environment rewards, demonstrating the superiority of our method.

REFERENCES

Ashutosh Adhikari, Xingdi Yuan, Marc-Alexandre Côté, Mikuláš Zelinka, Marc-Antoine Rondeau, Romain Laroche, Pascal Poupart, Jian Tang, Adam Trischler, and Will Hamilton. Learning dynamic belief graphs to generalize on text-based games. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 3045–3057, 2020.

Leonard Adolphs and Thomas Hofmann. Ledeepchef: Deep reinforcement learning agent for families of text-based games. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 34, pp. 7342–7349, 2020.

Alekh Agarwal, Nan Jiang, Sham M Kakade, and Wen Sun. Reinforcement learning: Theory and algorithms. *CS Dept., UW Seattle, Seattle, WA, USA, Tech. Rep*, 2019.

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

Prithviraj Ammanabrolu and Mark Riedl. Playing text-adventure games with graph-based deep reinforcement learning. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, volume 1, pp. 3557–3565, 2019a. doi: 10.18653/v1/N19-1358. URL `https://aclanthology.org/N19-1358`.

Prithviraj Ammanabrolu and Mark Riedl. Transfer in deep reinforcement learning using knowledge graphs. In *Proceedings of the Thirteenth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-13)*, pp. 1–10, 2019b. doi: 10.18653/v1/D19-5301. URL `https://aclanthology.org/D19-5301`.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, pp. 5048–5058, 2017.

Timothy Atkinson, Hendrik Baier, Tara Copplestone, Sam Devlin, and Jerry Swan. The text-based adventure ai competition. *IEEE Transactions on Games*, 11(3):260–266, 2019. doi: 10.1109/TG.2019.2896017.

Andrew G Barto. Intrinsic motivation and reinforcement learning. In *Intrinsically motivated learning in natural and artificial systems*, pp. 17–47. Springer, 2013.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. *Advances in neural information processing systems*, 29:1471–1479, 2016.

Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A Efros. Large-scale study of curiosity-driven learning. *arXiv preprint arXiv:1808.04355*, 2018a.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018b.

Andres Campero, Roberta Raileanu, Heinrich Küttler, Joshua B Tenenbaum, Tim Rocktäschel, and Edward Grefenstette. Learning with amigo: Adversarially motivated intrinsic goals. *arXiv preprint arXiv:2006.12122*, 2020.

Paul Christiano, Jan Leike, Tom B Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *arXiv preprint arXiv:1706.03741*, 2017.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, et al. Textworld: A learning environment for text-based games. *arXiv preprint arXiv:1806.11532*, 2018.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

Nancy Fulda, Daniel Ricks, Ben Murdoch, and David Wingate. What can you do with a rock? affordance extraction via word embeddings. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pp. 1039–1045, 2017. doi: 10.24963/ijcai.2017/144. URL https://doi.org/10.24963/ijcai.2017/144.

Abhishek Gupta, Benjamin Eysenbach, Chelsea Finn, and Sergey Levine. Unsupervised meta-learning for reinforcement learning. *arXiv preprint arXiv:1806.04640*, 2018.

Matthew Hausknecht, Ricky Loynd, Greg Yang, Adith Swaminathan, and Jason D Williams. Nail: A general interactive fiction agent. *arXiv preprint arXiv:1902.04259*, 2019.

Matthew Hausknecht, Prithviraj Ammanabrolu, Marc-Alexandre Côté, and Xingdi Yuan. Interactive fiction games: A colossal adventure. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 34, pp. 7903–7910, 2020.

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

Vishal Jain, William Fedus, Hugo Larochelle, Doina Precup, and Marc G Bellemare. Algorithmic improvements for deep reinforcement learning applied to interactive fiction. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 34, pp. 4328–4336, 2020.

Yiding Jiang, Shixiang Shane Gu, Kevin P Murphy, and Chelsea Finn. Language as an abstraction for hierarchical deep reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, pp. 9419–9431, 2019.

Leslie Pack Kaelbling. Learning to achieve goals. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1094–1098, 1993.

Alexander S Klyubin, Daniel Polani, and Chrystopher L Nehaniv. Empowerment: A universal agent-centric measure of control. In *2005 ieee congress on evolutionary computation*, volume 1, pp. 128–135. IEEE, 2005.

Bartosz Kostka, Jaroslaw Kwiecieli, Jakub Kowalski, and Pawel Rychlikowski. Text-based adventures of the golovin ai agent. In *2017 IEEE Conference on Computational Intelligence and Games (CIG)*, pp. 181–188. IEEE, 2017.

Nicolas Lair, Cédric Colas, Rémy Portelas, Jean-Michel Dussoux, Peter Ford Dominey, and Pierre-Yves Oudeyer. Language grounding through social interactions and curiosity-driven multi-goal learning. *arXiv preprint arXiv:1911.03219*, 2019.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. In *NIPS Deep Learning Workshop*, 2013.

Keerthiram Murugesan, Mattia Atzeni, Pushkar Shukla, Mrinmaya Sachan, Pavan Kapanipathi, and Kartik Talamadupula. Enhancing text-based reinforcement learning agents with commonsense knowledge. *arXiv preprint arXiv:2005.00811*, 2020.

Keerthiram Murugesan, Mattia Atzeni, Pavan Kapanipathi, Pushkar Shukla, Sadhana Kumaravel, Gerald Tesauro, Kartik Talamadupula, Mrinmaya Sachan, and Murray Campbell. Text-based rl agents with commonsense knowledge: New challenges, environments and baselines. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 35, pp. 9018–9027, 2021.

Karthik Narasimhan, Tejas D Kulkarni, and Regina Barzilay. Language understanding for text-based games using deep reinforcement learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1–11, 2015. doi: 10.18653/v1/D15-1001. URL https://aclanthology.org/D15-1001.

Georg Ostrovski, Marc G Bellemare, Aäron Oord, and Rémi Munos. Count-based exploration with neural density models. In *International conference on machine learning*, pp. 2721–2730. PMLR, 2017.

Pierre-Yves Oudeyer and Frederic Kaplan. What is intrinsic motivation? a typology of computational approaches. *Frontiers in neurorobotics*, 1:6, 2009.

Pierre-Yves Oudeyer, Frdric Kaplan, and Verena V Hafner. Intrinsic motivation systems for autonomous mental development. *IEEE transactions on evolutionary computation*, 11(2):265–286, 2007.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pp. 2778–2787. PMLR, 2017.

Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.

Vitchyr H Pong, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine. Skew-fit: State-covering self-supervised reinforcement learning. *arXiv preprint arXiv:1903.03698*, 2019.

Roberta Raileanu and Tim Rocktäschel. Ride: Rewarding impact-driven exploration for procedurally-generated environments. *arXiv preprint arXiv:2002.12292*, 2020.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International conference on machine learning*, pp. 1312–1320, 2015.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference (ESWC)*, pp. 593–607, 2018.

Jürgen Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*, pp. 222–227, 1991.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Laura Schulz. Finding new facts; thinking new thoughts. *Advances in child development and behavior*, 43:269–294, 2012.

Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. In *International Conference on Learning Representations*, 2019.

Kyriacos Shiarlis, Markus Wulfmeier, Sasha Salter, Shimon Whiteson, and Ingmar Posner. Taco: Learning task decomposition via temporal alignment for control. In *International Conference on Machine Learning (ICML)*, volume 80, pp. 4654–4663, 2018.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Satinder Singh, Richard L Lewis, and Andrew G Barto. Where do rewards come from. In *Proceedings of the annual conference of the cognitive science society*, pp. 2601–2606. Cognitive Science Society, 2009.

Sungryull Sohn, Junhyuk Oh, and Honglak Lee. Hierarchical reinforcement learning for zero-shot generalization with subtask dependencies. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, pp. 7156–7166, 2018.

Jonathan Sorg, Satinder P Singh, and Richard L Lewis. Internal rewards mitigate agent boundedness. In *ICML*, 2010.

Bradly C Stadie, Sergey Levine, and Pieter Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv preprint arXiv:1507.00814*, 2015.

Alexander L Strehl and Michael L Littman. An analysis of model-based interval estimation for markov decision processes. *Journal of Computer and System Sciences*, 74(8):1309–1331, 2008.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 23–30. IEEE, 2017.

Jack Urbanek, Angela Fan, Siddharth Karamcheti, Saachi Jain, Samuel Humeau, Emily Dinan, Tim Rocktäschel, Douwe Kiela, Arthur Szlam, and Jason Weston. Learning to speak and act in a fantasy text adventure game. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 673–683, 2019. doi: 10.18653/v1/D19-1062. URL `https://aclanthology.org/D19-1062`.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, pp. 5998–6008, 2017.

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

Xusen Yin and Jonathan May. Comprehensible context-driven text game playing. *IEEE Conference on Games (CoG)*, pp. 1–8, 2019.

Xusen Yin, Ralph Weischedel, and Jonathan May. Learning to generalize for sequential decision making. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing: Findings (EMNLP-Findings)*, pp. 3046–3063, 2020. doi: 10.18653/v1/2020.findings-emnlp.273. URL `https://aclanthology.org/2020.findings-emnlp.273`.

Xingdi (Eric) Yuan, Marc-Alexandre Côté, Alessandro Sordoni, Romain Laroche, Remi Tachet des Combes, Matthew Hausknecht, and Adam Trischler. Counting to explore and generalize in text-based games. In *European Workshop on Reinforcement Learning (EWRL)*, 2018.

Tom Zahavy, Matan Haroush, Nadav Merlis, Daniel J Mankowitz, and Shie Mannor. Learn what not to learn: Action elimination with deep reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, pp. 3562–3573, 2018.

Mikulas Zelinka, Xingdi Yuan, Marc-Alexandre Côté, Romain Laroche, and Adam Trischler. Building dynamic knowledge graphs from text-based games. *arXiv preprint arXiv:1910.09532*, 2019.

Tianjun Zhang, Huazhe Xu, Xiaolong Wang, Yi Wu, Kurt Keutzer, Joseph E Gonzalez, and Yuandong Tian. Bebold: Exploration beyond the boundary of explored regions. *arXiv preprint arXiv:2012.08621*, 2020.

Zeyu Zheng, Junhyuk Oh, and Satinder Singh. On learning intrinsic rewards for policy gradient methods. *arXiv preprint arXiv:1804.06459*, 2018.

A Examples of Text-based Games

Figure 4 visualizes the initial observation o_0^{KG} for four games, where "S1 Game1" and "S1 Game2" belong to level "S1", "S2 Game1" and "S2 Game2" belong to level "S2". Figure 5 visualizes the initial observation of two games belonging to level "S3". Figure 6 visualizes the initial observation of one game belonging to level "S4". Games within the same level have the same complexity, but are different in their layouts. For example, "S2 Game1" and "S2 Game2" have the same number of rooms, number of ingredients and number of requirements, but the ingredient and requirement are different. Similarly, "S3 Game1" and "S3 Game2" have the same number of rooms, but are with different room connectivity. Games within different levels have different complexities, and their layouts are naturally different (e.g., "S1 Game1" v.s., "S2 Game1" v.s., "S3 Game1" v.s., "S4 Game1"). The TextWorld also provides other game themes, such as the default House theme, the Coin Collector theme and the Treasure Hunter theme. In these themes, the agent needs to go through the rooms to find either the coin, or an object specified at the beginning of an episode. We believe that the Cooking theme we use in this work is able to cover these themes, as it requires the agent to navigate through different numbers of rooms, collect different numbers of ingredients, and prepare them in different ways.

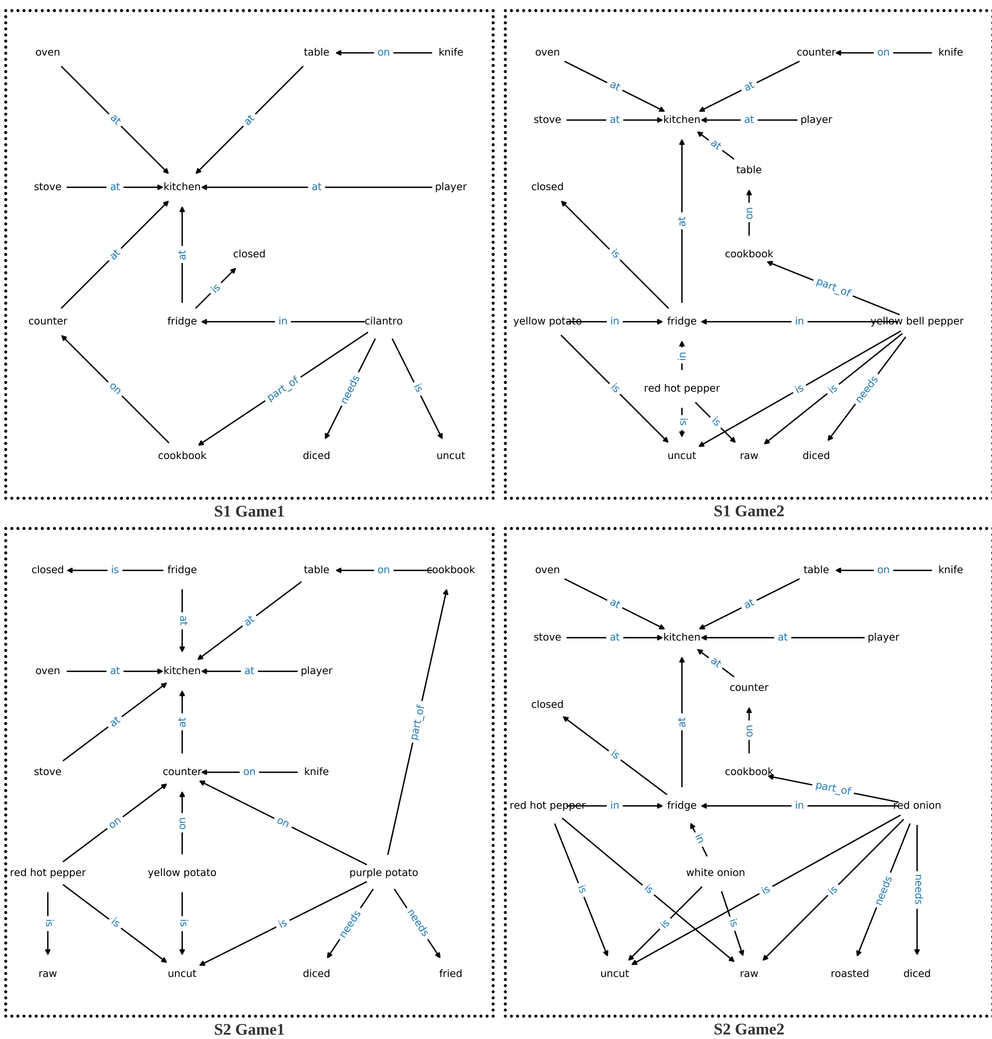


Figure 4: The initial observation of four games, where "S1 Game1" and "S1 Game2" belong to level "S1", "S2 Game1" and "S2 Game2" belong to level "S2".

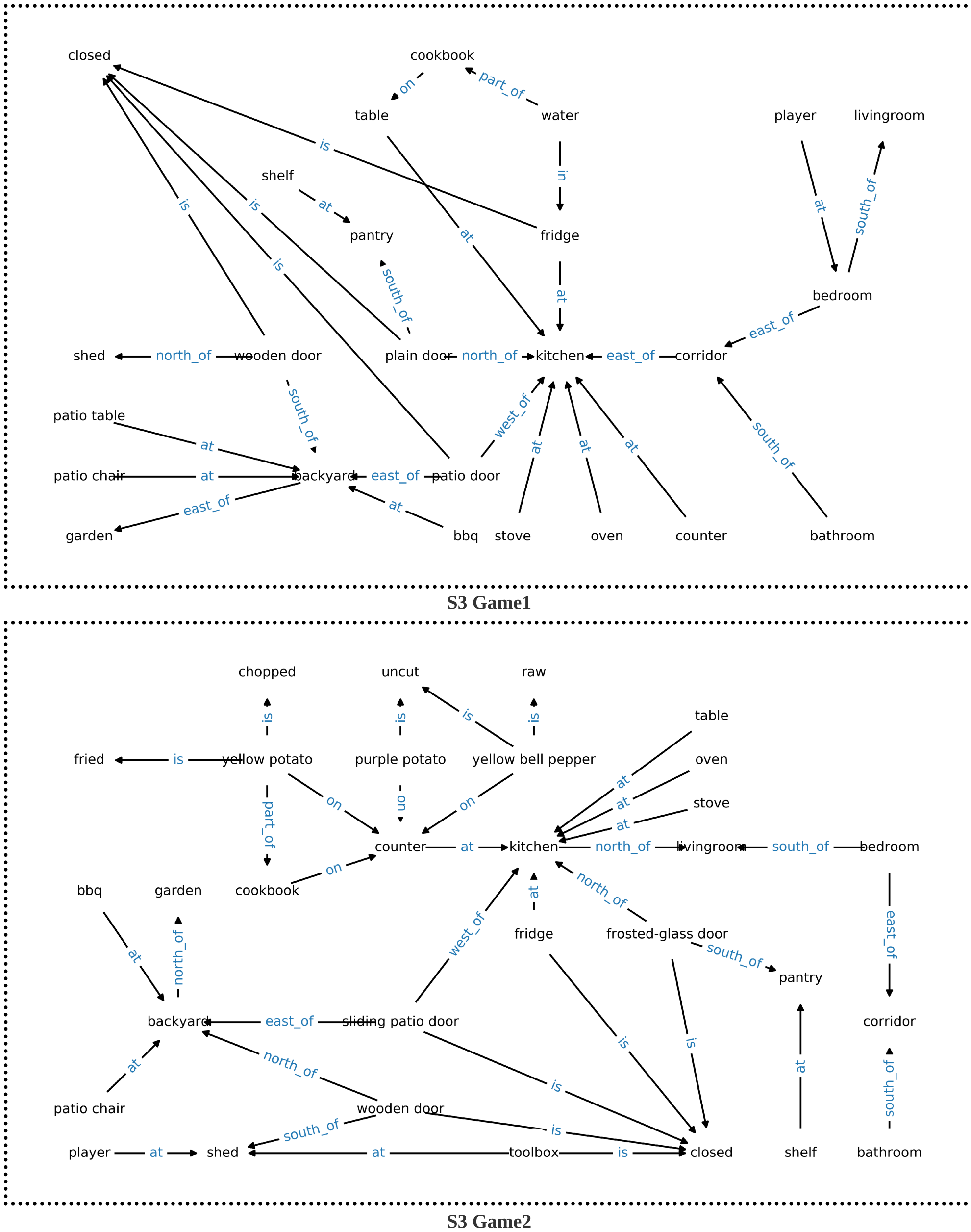
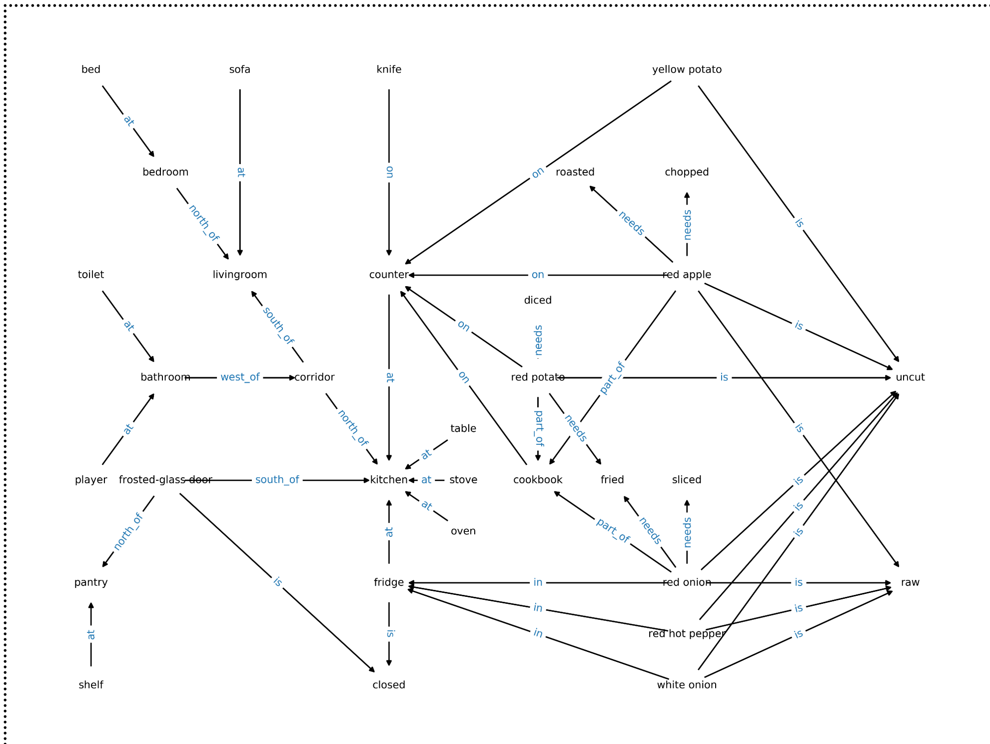


Figure 5: The initial observation of two games belonging to level "S3".

B Goal set generator

We design a simple non-learning-based goal set generator by exploiting the KG-based observation in the cooking theme. Algorithm 2 shows the pipeline for obtaining the goal set $\mathcal{G}_t$. We first obtain the ingredient set $\mathcal{I}$. For each ingredient $i \in \mathcal{I}$, we first check whether it has been collected, then obtain its status set $\mathcal{S}_i$ and requirement set $\mathcal{R}_i$. We consider three types of goals: 1) "find" requires the agent to find and collect an uncollected ingredient, 2) "prepare" requires the agent to prepare an ingredient to satisfy a requirement, and 3) "eat", that the agent is required to prepare and eat the final meal. Algorithm 3 shows the pipeline for assigning the goal-conditioned reward r_t^{goal}. We first obtain the type of a goal g, then check whether this goal has been accomplished given a_t and o_{t+1}^{KG}. Some functions in Algorithm 2 can be reused here. r_t^{goal} is a binary reward that we will assign



S4 Game1

Figure 6: The initial observation of one game belonging to level “S4”.

$r_t^{\text{goal}} = r_{\text{max}}$ if g is accomplished successfully, otherwise r_{min} (still not finished, or failed). Algorithm 2 and Algorithm 3 can also be implemented via learning-based methods. For example, the functions can be achieved by a QA model by answering specific questions.

Algorithm 2 Goal set generation

Input: Knowledge graph o_t^{KG}
Initialize: Goal set $\mathcal{G}_t \leftarrow \emptyset$
Get ingredient set $\mathcal{I} \leftarrow \text{GetIng}(o_t^{\text{KG}})$
for each ingredient $i \in \mathcal{I}$ **do**
if CheckCollection$(i, o_t^{\text{KG}}) = \text{False}$ **then**
Add goal "find i" to $\mathcal{G}_t$
else
Get status set $\mathcal{S}_i \leftarrow \text{GetStatus}(i, o_t^{\text{KG}})$
Get requirement set $\mathcal{R}_i \leftarrow \text{GetReq}(i, o_t^{\text{KG}})$
for each requirement $r_i \in \mathcal{R}_i$ **do**
if $r_i \notin \mathcal{S}_i$ **then**
Add goal "r_i i" to $\mathcal{G}_t$
end if
end for
end if
end for
if $\mathcal{G}_t = \emptyset$ **then**
Add goal "prepare and eat meal" to $\mathcal{G}_t$
end if
return $\mathcal{G}_t$.

Algorithm 3 Goal-conditioned reward acquisition

Input: Knowledge graph o_{t+1}^{KG}, goal g, rewards $\{r_{\text{min}}, r_{\text{max}}\}$
Initialize: FLAG $\leftarrow$ False
Obtain goal type $g_{\text{type}} \leftarrow \text{GetGoalType}(g)$
if $g_{\text{type}} =$ "find" **then**
Get ingredient i from g
if CheckCollection$(i, o_{t+1}^{\text{KG}}) = \text{True}$ **then**
FLAG $\leftarrow$ True
end if
else if $g_{\text{type}} =$ "prepare" **then**
Get ingredient i, requirement r from g
Get status set $\mathcal{S}_i \leftarrow \text{GetStatus}(i, o_{t+1}^{\text{KG}})$
if $r \in \mathcal{S}_i$ **then**
FLAG $\leftarrow$ True
end if
else
if CheckExistance("meal", $o_{t+1}^{\text{KG}}) = \text{True}$ **then**
FLAG $\leftarrow$ True
end if
end if
if FLAG $=$ True **then**
return r_{max}.
else
return r_{min}.
end if