Generalization in Text-based Games via Hierarchical Reinforcement Learning

Yunqiu Xu[1], **Meng Fang**[2], **Ling Chen**[1], **Yali Du**[3], **Chengqi Zhang**[1]
[1]Australian Artificial Intelligence Institute, University of Technology Sydney, Sydney, Australia
Yunqiu.Xu@student.uts.edu.au, {Ling.Chen,Chengqi.Zhang}@uts.edu.au
[2]Eindhoven University of Technology, Eindhoven, the Netherlands
m.fang@tue.nl
[3]King's College London, London, United Kingdom
yali.du@kcl.ac.uk

Abstract

Deep reinforcement learning provides a promising approach for text-based games in studying natural language communication between humans and artificial agents. However, the generalization still remains a big challenge as the agents depend critically on the complexity and variety of training tasks. In this paper, we address this problem by introducing a hierarchical framework built upon the knowledge graph-based RL agent. In the high level, a meta-policy is executed to decompose the whole game into a set of subtasks specified by textual goals, and select one of them based on the KG. Then a sub-policy in the low level is executed to conduct goal-conditioned reinforcement learning. We carry out experiments on games with various difficulty levels and show that the proposed method enjoys favorable generalizability.

1 Introduction

Text-based games are simulated systems where the agent takes textual observations as the input, and interacts with the environment via text commands (Hausknecht et al., 2020). They are suitable test-beds to study natural language understanding, commonsense reasoning and language-informed decision making (Luketina et al., 2019). Reinforcement Learning (RL) based agents (Narasimhan et al., 2015; Zahavy et al., 2018) have been developed to handle challenges such as language-based representation learning and combinatorial action space. Among them, KG-based agents (Ammanabrolu and Hausknecht, 2020) yield promising performance with the aid of Knowledge Graph (KG), which serves as a belief state to provide structural information.

To design intelligent RL-based agents for text-based games, it is necessary to build agents that automatically learn to solve different games. However, generalization remains as one of the key challenges of RL − the agent tends to overfit the training environment and fails to generalize to new environments (Cobbe et al., 2019). In the domain of text-based games, the TextWorld (Côté et al., 2018) makes it feasible to study generalizability by creating non-overlapping game sets with customizable domain gaps (e.g., themes, vocabulary sets, difficulty levels and layouts). Most previous works study generalizability either upon games with the same difficulty level but different layouts (Ammanabrolu and Riedl, 2019a), or upon games with a set of multiple levels that have been observed during training (Adolphs and Hofmann, 2020). Although these agents perform well on relatively simple games, they can hardly achieve satisfactory performance on difficult games (Adhikari et al., 2020). In this work, we aim to develop agents that can be generalized to not only games from the same difficulty level while having unseen different layouts, but also games from unseen difficulty levels where both layouts and complexities are different.

While solving a whole game might be difficult due to long-term temporal dependencies, and the learnt strategy might be difficult to be transferred to other games due to large domain gaps, it would be more flexible to treat the game as a sketch of subtasks (Andreas et al., 2017; Oh et al., 2017). This brings two branches of benefits. First, the subtasks would be easier to solve as they have short-term temporal dependencies. Second, the strategies learnt for solving subtasks may be recomposed to solve an unseen game. Motivated by these insights, we aim to solve a game by decomposing it into subtasks characterized by textual goals, then making decisions conditioned on them. Instead of handcrafting the task sketches, we leverage the hierarchical reinforcement learning (HRL) framework for adaptive goal selection, and exploit the compositional nature of language (Jiang et al., 2019) to improve generalizability.

We develop a two-level framework, **H**ierarchical

Findings of the Association for Computational Linguistics: EMNLP 2021, pages 1343–1353
November 7–11, 2021.

Knowledge **Graph**-based **A**gent (H-KGA)[1], to learn a hierarchy of policies with the aid of KG. In the high level, we use a meta-policy to obtain a set of available goals characterized by texts, and select one of them according to the current KG-based observation. Then, we use a sub-policy for goal-conditioned reinforcement learning. Besides, we design a scheduled training strategy to facilitate learning across multiple levels. We conduct experiments on a series of cooking games across 8 levels, while only 4 levels are available during training. The experimental results show that our method improves generalizability in both seen and unseen levels.

Our contributions are summarised as follows: Firstly, we are the first to study generalizability in text-based games from the aspect of hierarchical reinforcement learning. Secondly, we develop a two-level HRL framework leveraging the KG-based observation for adaptive goal selection and goal-conditioned decision making. Thirdly, we empirically validate the effectiveness of our method in games with both seen and unseen difficulty levels, which show favorable generalizability.

2 Related work

2.1 RL agent for text-based games

Motivated by the prosperity of deep reinforcement learning techniques in playing games (Silver et al., 2016), robotics (Schulman et al., 2017; Fang et al., 2019a,b) and NLP (Fang et al., 2017), several RL-based game agents have been developed for text-based games (He et al., 2016; Yuan et al., 2018; Jain et al., 2020; Yin and May, 2019; Guo et al., 2020; Xu et al., 2020a). Compared with the non-learning-based agents (Hausknecht et al., 2019; Atkinson et al., 2019), the RL-based agents are more favorable as there is no need to handcraft game playing strategies with huge amounts of expert knowledge. The KG-based agents (Murugesan et al., 2020; Xu et al., 2020b) extend RL-based agents with the knowledge graph, which can be constructed from the raw textual observation via simple rules (Ammanabrolu and Riedl, 2019a), language models (Ammanabrolu et al., 2020) or pre-training tasks (Adhikari et al., 2020). The major benefit of KG is that it serves as a belief state to provide structural and historical information to handle partial observability. While these works focus on constructing KG from the textual observation, we aim at improving generalizability by fully exploiting the KG to design a goal-conditioned HRL. Our work thus complements KG-based agents.

2.2 Generalization in text-based games

It may be 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 and 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 and Riedl, 2019a; Murugesan et al., 2021), or a mixture of multiple levels (Adolphs and Hofmann, 2020; Yin et al., 2020), or both (Adhikari et al., 2020). In this work, we extend the setting of multiple levels to unseen levels. We not only study generalization in games that have the same difficulty level but various layouts, but also consider games where both the layouts and levels are different from those of the training games. In addition, we emphasize on improving the performance on hard levels.

2.3 Hierarchical reinforcement learning

The HRL framework (Dayan and Hinton, 1992) has been studied in video games (Kulkarni et al., 2016; Vezhnevets et al., 2017; Shu et al., 2018), robotic control tasks (Nachum et al., 2018),and NLP tasks such as the dialogue system (Peng et al., 2017; Saleh et al., 2020). However, as far as we know, we are the first to introduce the insight into text-based games with KG-based observation. Previous works also considered identifying a task by textual goal specifications (Bahdanau et al., 2019; Fu et al., 2019). In the domain of text-based games, such goal-conditioned RL setting has been studied with the quest generation tasks (Ammanabrolu et al., 2019, 2021). In our work, we specify a subtask by its goal. Different from these works, where a single goal is pre-specified or directly generated from the observation, we introduce a hierarchy by

[1]Code is available at: https://github.com/YunqiuXu/H-KGA

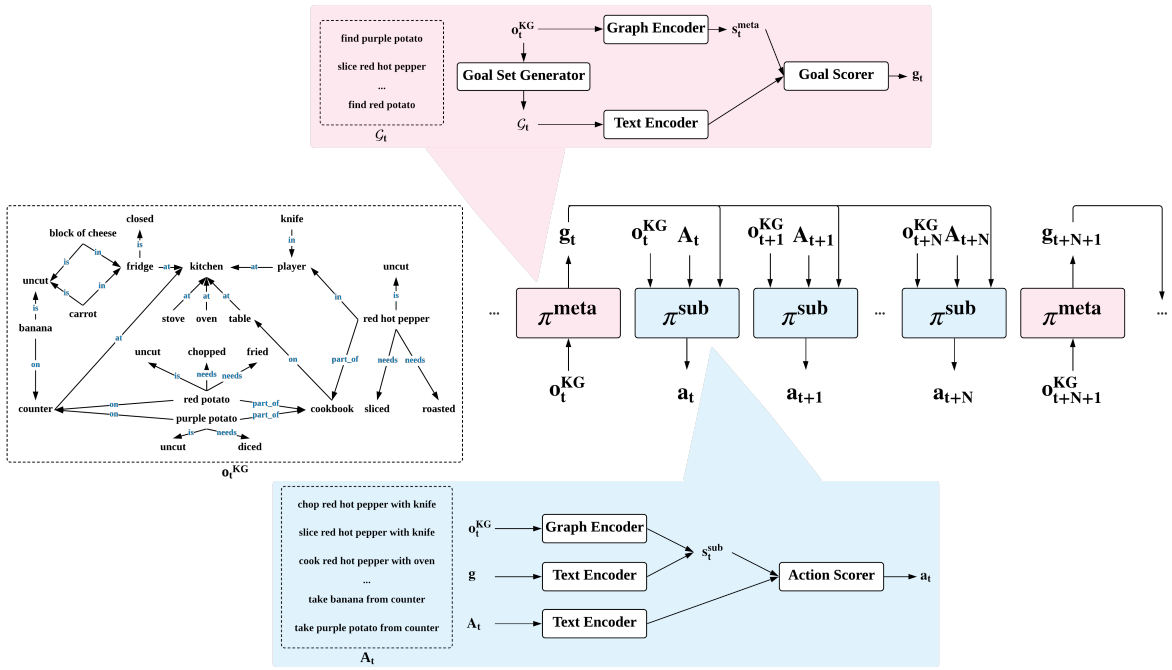

Figure 1: The overview of H-KGA. In the high level (red), the meta-policy π^{meta} first obtains the set of goals of available subtasks $\mathcal{G}_t$ from o_t^{KG}, then selects a goal g_t for the sub-policy π^{sub}. In the low level (blue), π^{sub} selects the action a_t from the admissible action set A_t conditioned on o_t^{KG} and g.

disentangling the process of goal set generation and goal selection. By accommodating flexible goal set generation (e.g., by pre-trained language models or human experts), we focus on designing a meta-policy to select the goal in an adaptive manner. By adopting HRL to select a textual-based goal for the sub-policy, our work is similar to HIN (Jiang et al., 2019) which, however, focuses on visual scenarios and separately trains the meta-policy and the sub-policy, leaving joint training as future work. Instead, we consider the domain of text-based games, and develop a framework to enable joint training of meta-policy and sub-policy. We further compare joint and individual training in Sec. 6.

3 Background

3.1 KG-based observation

Following previous works (Hausknecht et al., 2020), we formulate the text-based games as Partially Observable Markov Decision Processes (POMDPs), where the details is in Appendix A. We discard the raw textual observation and consider only the KG-based observation o_t^{KG} as the observational input at timestep t. Fig. 1 shows an example of o_t^{KG}. The KG is defined as $G = (V, E)$, where V and E are the node set and the edge set, respectively. o_t^{KG} consists of a set of triplets, where a triplet is formulated as $\langle Subject, Relation, Object \rangle$, denoting that the $Subject \in V$ has $Relation \in E$ with the $Object \in V$.

3.2 Problem setting

We aim to design an RL-based agent that is able to address the generalization in solving text-based games. To reduce the requirement for external knowledge, we consider games sharing similar themes and vocabularies, but varying 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. To accomplish it, the player has to collect ingredients and prepare them in correct ways. 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. We follow the multi-task learning setting to consider that the training set and the testing set consist of multiple games from multiple levels. We consider two scenarios of generalization: 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. More examples are provided in Appendix. D.

4 Methodology

4.1 Overview

Fig. 1 shows the overview of H-KGA, which consists of a hierarchy of two levels of policies. In the high level, a meta-policy π^{meta} first obtains the set of goals of available subtasks $\mathcal{G}_t$ from o_t^{KG}, then selects a goal g_t for the sub-policy π^{sub}. In the low level, π^{sub} selects the action a_t from the admissible action set A_t conditioned on o_t^{KG} and g. We omit the subscript "t" for g because this goal may be selected in the past rather than the current time step. For example, as shown in Fig. 1, once the g_t is selected by π^{meta}, it will remain unchanged for N time steps until being completed (e.g., accomplished or failed). At each time step from t to $t+N$, π^{sub} considers the same goal g_t.

In the following, we illustrate how to design π^{meta} to obtain the available goal set $\mathcal{G}_t$ and conduct goal selection to obtain $g_t \in \mathcal{G}_t$ in Sec. 4.2; how to design π^{sub} to select an action $a_t \in A_t$ in Sec. 4.3; and how to train H-KGA with a scheduled curriculum for multi-task learning in Sec. 4.4.

4.2 Meta-policy for goal selection

As discussed before, while a whole game may be difficult to accomplish due to long-term temporal dependency, decomposing it into a sketch of 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. Therefore, inspired by the HRL framework (Sutton et al., 1999), we design a meta-policy π^{meta} to first obtain a set of subtasks, then select one subtask from them. We characterize a subtask by its goal to transform subtask selection into goal selection. 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 (Andrychowicz et al., 2017). Fig. 1 shows the overview of π^{meta} (in red), which consists of a goal set generator, a graph encoder, a text encoder and a goal scorer. We denote the set containing all required goals for solving a game as $\mathcal{G}$. Then we define a goal as "available" at a time step if no other goals should be accomplished before it. For example, "cook red potato" is not available in Fig. 1, as another goal "find red potato" should be accomplished first. The goal set generator has two purposes: 1) obtain the set of currently available goals $\mathcal{G}_t \subseteq \mathcal{G}$, and 2) check whether a goal has been accomplished. 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, we use a non-learning-based method to obtain $\mathcal{G}_t$ and the details are discussed in Sec 5.3 and Appendix B.

After obtaining $\mathcal{G}_t$, π^{meta} will be used to select a goal $g_t \in \mathcal{G}_t$. We use a graph encoder to encode o_t^{KG} as state representation $\boldsymbol{s}_t^{\text{meta}}$, and a text encoder to encode $\mathcal{G}_t$ as a stack of goal representations. Arbitrary graph encoders and text encoders can be used. We implement 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. In the goal scorer, we adopt a goal scoring process similar to (He et al., 2016), where $\boldsymbol{s}_t^{\text{meta}}$ will be paired with each goal representation, then processed by linear layers to obtain the goal scores. The scores can be treated as either sampling probabilities or Q values, where the goal candidate with the highest Q value will be selected.

Following the Semi-Markov Decision Process (SMDPs) (Sutton et al., 1999), π^{meta} will be re-executed once a goal is accomplished / failed. π^{meta} receives rewards r_t^{env} from the environment. In a transition for π^{meta}, the reward is set as the sum of environment rewards:

$$r^{\text{meta}} = \sum_{i=1}^{T} r_{t+i}^{\text{env}} \tag{1}$$

where T denotes time steps for accomplishing g_t.

4.3 Sub-policy for action selection

The sub-policy π^{sub} follows the goal-conditioned RL setting (Kaelbling, 1993) where a_t is selected by considering both o_t^{KG} and g. Fig. 1 shows the architecture of π^{sub} (in blue), which is similar to π^{meta} except that the state $\boldsymbol{s}_t^{\text{sub}}$ is constructed based on both o_t^{KG} and g. The graph encoder and text encoder in π^{meta} can be re-used in π^{sub}, or be re-initialized with new weights. 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. We denote an action as "admissible" if it does not lead to meaningless feedback (e.g., "Nothing happens"). Similar to the goal scorer in π^{meta}, the action scorer will pair $\boldsymbol{s}_t^{\text{sub}}$ with each candidate $a_i \in A_t$, followed by linear layers to compute the action scores.

Depending on goal accomplishment, π^{sub} receives binary intrinsic reward $r_t^{\text{goal}} \in \{r_{\text{min}}, r_{\text{max}}\}$, which in this work can be determined by reusing the goal set generator upon o_{t+1}^{KG}. Take Fig. 1 as an example. If the goal before observing o_t^{KG} is "find knife", the agent will receive $r_t^{\text{goal}} = r_{\text{max}}$, as this goal is accomplished at time step t. Although the KG can serve as a "map" to provide guidance, such binary reward is insufficient for the agent to accomplish a goal in complex games (e.g., the agent has to go through multiple rooms to find an ingredient). To further improve the performance of π^{sub}, we reshape the sub-reward with the count-based intrinsic reward (Bellemare et al., 2016) to encourage exploration. Specifically, we apply the BeBold method (Zhang et al., 2020) to the text-based games domain. During training, we count the visitation of observations within an episode, and the accumulated visitation throughout the training process. The count-based reward r_t^{count} is then defined as the regulated difference of inverse cumulative visitation counts with episodic restriction:

$$r_{t+1}^{\text{count}} = \max\Big(\frac{1}{\text{N}_{\text{acc}}(o_t^{\text{KG}})} - \frac{1}{\text{N}_{\text{acc}}(o_{t+1}^{\text{KG}})}, 0\Big) \cdot \mathbb{I}\{\text{N}_{\text{epi}}(o_{t+1}^{\text{KG}}) = 1\} \quad (2)$$

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. The reward for π^{sub} can then be obtained by combining r_{t+1}^{goal} and r_{t+1}^{count}:

$$r_{t+1}^{\text{sub}} = r_{t+1}^{\text{goal}} + \lambda \cdot r_{t+1}^{\text{count}} \quad (3)$$

where λ is a constant coefficient.

4.4 Training H-KGA for multi-task learning

We train H-KGA via Double DQN (Hasselt et al., 2016) with prioritized experience replay (Schaul et al., 2015). Algo. 1 shows the training strategy. We consider a training set $\mathcal{D}_{\text{train}}$ with $\mathcal{L}$ levels of games. For each episode, we sample a game x from $\mathcal{D}_{\text{train}}$ to interact with (lines 2-22). A goal g will be terminated if it is accomplished/ failed, or

Algorithm 1 Training Strategy for H-KGA

Input: game sets $\{\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{val}}\}$, replay buffers $\{B^{\text{meta}}, B^{\text{sub}}\}$, update frequencies $\{F_{\text{up}}^{\text{meta}}, F_{\text{up}}^{\text{sub}}\}$, validation frequency F_{val}, tolerance τ, coefficients β, λ, patience P
Initialize: counters $k \leftarrow 1$, $p \leftarrow 0$, $N_{\text{acc}} \leftarrow \emptyset$, $N_{\text{epi}} \leftarrow \emptyset$, best validation score $V_{\text{val}} \leftarrow 0$, $r^{\text{meta}} \leftarrow 0$, caches $\{C^{\text{meta}}, C^{\text{sub}}\}$, policies $\{\pi^{\text{meta}}, \pi^{\text{sub}}\}$, $\{\Pi^{\text{meta}}, \Pi^{\text{sub}}\}$

1: **for** $e \leftarrow 1$ to NUM_EPISODES **do**
2: $l \leftarrow$ SampleLevel$(\mathcal{L}, p_l)$
3: $x \leftarrow$ SampleGame$(\mathcal{D}_{\text{train}}, l)$
4: $o_0^{\text{KG}} \leftarrow$ reset x
5: $C^{\text{meta}} \leftarrow \emptyset$, $C^{\text{sub}} \leftarrow \emptyset$, $N_{\text{epi}} \leftarrow \emptyset$,
6: Update N_{acc}, N_{epi} with o_0^{KG}
7: **for** $t \leftarrow 0$ to NUM_STEPS **do**
8: $g \leftarrow \pi^{\text{meta}}(g|o_t^{\text{KG}})$
9: $r^{\text{meta}} \leftarrow 0$
10: **while** g is not terminated **do**
11: $a_t \leftarrow \pi^{\text{sub}}(a|o_t^{\text{KG}}, g)$
12: Execute a_t, receive o_{t+1}^{KG}, r_{t+1}^{env}, obtain r_{t+1}^{goal}
13: Update N_{acc}, N_{epi} with o_{t+1}^{KG}
14: Compute r_{t+1}^{sub} using Eq. (2) and Eq. (3)
15: Store the sub transition into C^{sub}
16: $r^{\text{meta}} \leftarrow r^{\text{meta}} + r_{t+1}^{\text{env}}$
17: $t \leftarrow t + 1$
18: $k \leftarrow k + 1$
19: **if** $k\%F_{\text{up}}^{\text{meta}} = 0$ **then**
20: Update$(\pi^{\text{meta}}, B^{\text{meta}})$
21: **if** $k\%F_{\text{up}}^{\text{sub}} = 0$ **then**
22: Update$(\pi^{\text{sub}}, B^{\text{sub}})$
23: Store the meta transition into C^{meta}
24: Update p_l using Eq. (4)
25: **if** Avg$(r^{\text{meta}}|C^{\text{meta}}, l) > \tau \cdot$ Avg$(r^{\text{meta}}|B^{\text{meta}}, l)$ **then**
26: Store all transitions in C^{meta} into B^{meta}
27: **if** Avg$(r^{\text{goal}}|C^{\text{sub}}, l) > \tau \cdot$ Avg$(r^{\text{goal}}|B^{\text{sub}}, l)$ **then**
28: Store all transitions in C^{sub} into B^{sub}
29: **if** $e\%F_{\text{val}} = 0$ **then**
30: $v_{\text{val}} \leftarrow$ Validate$(\pi^{\text{meta}}, \pi^{\text{sub}}, \mathcal{D}_{\text{val}})$
31: **if** $v_{\text{val}} \geq V_{\text{val}}$ **then**
32: $V_{\text{val}} \leftarrow v_{\text{val}}$, $\Pi^{\text{meta}} \leftarrow \pi^{\text{meta}}$, $\Pi^{\text{sub}} \leftarrow \pi^{\text{sub}}$
33: $p \leftarrow 0$, continue
34: **if** $p > P$ **then**
35: $\pi^{\text{meta}} \leftarrow \Pi^{\text{meta}}$, $\pi^{\text{sub}} \leftarrow \Pi^{\text{sub}}$, $p \leftarrow 0$
36: **else**
37: $p \leftarrow p + 1$

t exceeds NUM_STEPS. We formulate the meta transition as $\langle o_t^{\text{KG}}, g, r^{\text{meta}}, o_{t+T}^{\text{KG}}, l\rangle$, and the sub transition as $\langle (o_t^{\text{KG}}, g), a_t, r_{t+1}^{\text{sub}}, r_{t+1}^{\text{goal}}, (o_{t+1}^{\text{KG}}, g), l\rangle$, where $l \in \mathcal{L}$ denotes the level of a game. We update π^{meta} (π^{sub}) per $F_{\text{up}}^{\text{meta}}$ ($F_{\text{up}}^{\text{sub}}$) interaction steps, by sampling a batch of transitions from the replay buffer B^{meta} (B^{sub}). In addition, we leverage two strategies empirically effective for previous agents (Adhikari et al., 2020). First, we collect the episodic transitions within a cache, and only push them into the replay buffer when its average reward is greater than τ times the average reward of the buffer (lines 23-26). Second, we validate the model on a validation set $\mathcal{D}_{\text{val}}$ per F_{val} episodes and keep track of the best score V and the correspond-

ing policies $\{\Pi^{\text{meta}}, \Pi^{\text{sub}}\}$. We load the training policies $\{\pi^{\text{meta}}, \pi^{\text{sub}}\}$ back to $\{\Pi^{\text{meta}}, \Pi^{\text{sub}}\}$, if the validation performance v_{val} keeps being worse than V for over P times (lines 27-35).

The training process can be formulated as multi-task learning if we treat learning on games from the same level as a task. While the knowledge can be shared across levels, different levels may have different scales of training time and performance. For example, those from hard levels generally require more time to learn and tend to have lower normalized performance. To facilitate such multi-task learning setting, we further propose two strategies to improve Algorithm 1: 1) scheduled task sampling and 2) level-aware replay buffer. The scheduled task sampling is inspired by the curriculum learning (Bengio et al., 2009), where we schedule the tasks based on their difficulties. We track the training performance v_l on a level l, and compute the sampling probability as:

$$p_l = \frac{\exp(\beta - v_l)}{\sum_{l_i \in \mathcal{L}} \exp(\beta - v_{l_i})} \tag{4}$$

where β is a constant coefficient. For each episode, we first sample a level based on the probabilities, and then sample a training game from this level uniformly (lines 2-3). Compared to level-invariant sampling, this strategy encourages the agent to focus more on hard levels with training going on. Another strategy, level-aware replay buffer, is conducted when moving transitions from cache to the replay buffer (lines 23-26). As the transitions collected from hard games tend to have lower reward, they are not likely to be added to the replay buffer. To alleviate this problem, we make the level as an additional component of transition and record the average reward of each level. Then we compare those belonging to the same level to determine whether to add new transitions.

5 Experiments

5.1 Experiment setting

We conduct experiments on multiple levels of cooking games (Côté et al., 2018). 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.

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

5.2 Baselines

We consider the following five models, and compare with more variants in ablation studies:

- GATA (Adhikari et al., 2020): a powerful KG-based agent and the state-of-the-art on the rl.0.1 game set. However, it does not have hierarchical architecture, and the action selection policy is not goal-conditioned.
- GC-GATA: GATA equipped with a goal set generator, a **g**oal-**c**onditioned action selection (sub-)policy, and a non-learnable meta-policy for random goal selection.
- H-KGA: the proposed model with both meta-policy and sub-policy.
- H-KGA HalfJoint: an H-KGA variant, where during the first **half** of training process only the sub-policy is trained, then the two policies are **joint**ly trained.
- H-KGA Ind: an H-KGA variant, where the two policies are **ind**ividually trained (the sub-policy for the first half, then the meta-policy).

5.3 Implementation details

We implement the models based on GATA's released code[3]. In particular, we adopt the version

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

[3] https://github.com/xingdi-eric-yuan/GATA-public

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

Model	Avg Seen	Avg Unseen	Avg All
GATA	0.47±0.04	0.62±0.07	0.55±0.06
GC-GATA	0.54±0.13	0.61±0.12	0.58±0.12
H-KGA (ours)	**0.72**±0.04	**0.79**±0.04	**0.76**±0.03
H-KGA HalfJoint	0.56±0.11	0.57±0.07	0.57±0.09
H-KGA Ind	0.70±0.02	0.68±0.01	0.69±0.02
GATA w/o BeBold	0.54±0.06	0.68±0.02	0.61±0.03
H-KGA w/o BeBold	0.57±0.07	0.65±0.09	0.61±0.07
H-KGA w/o Sch	0.52±0.07	0.63±0.07	0.57±0.05
H-KGA w/o Sch w/o LR	0.63±0.14	0.63±0.16	0.63±0.15

GATA-GTF and denote it as GATA for simplicity. GATA-GTF discards textual observations and uses the ground truth full KG as observation, so that there is no information extraction error incurred during KG construction. We design a simple non-learning-based goal set generator to obtain available goals (leaving pre-training-based generators as future work). Please refer to Appendix B for details. All models follow the same architecture of graph encoder (i.e., R-GCNs), text encoder (i.e., single transformer block with single head) and scorers (i.e., linear layers). The encoders in π^{meta} and π^{sub} are initialized separately.

We set the step limit of an episode as 50 for training and 100 for validation / testing. We train the models for 100,000 episodes. All models apply the BeBold reward bonus with $\lambda = 0.1$, and the scheduled sampling method with $\beta = 1.0$. We set B^{meta} with size 50,000 and B^{sub} with size 500,000. We set $F_{\text{up}}^{\text{meta}}$ and $F_{\text{up}}^{\text{sub}}$ as 50 time steps, and the updating starts after 100 episodes with batch size 64. The GC-GATA pre-trained for 50,000 episodes is used for initializing H-KGA HalfJoint and H-KGA Ind. 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 $\mathcal{D}_{\text{test}}^{\text{unseen}}$. The experiments are conducted on a Quadro RTX 6000 GPU. Each experiment is run with 3 random seeds, and each run takes 2-3 days to finish.

5.4 Evaluation metrics

We denote a game's score as the episodic sum of rewards without discount. We use the normalized score, which is defined as the collected score normalized by the maximum available score for this game, to measure the performance. For each testing game set, we report the performance on each level and the performance averaged over levels.

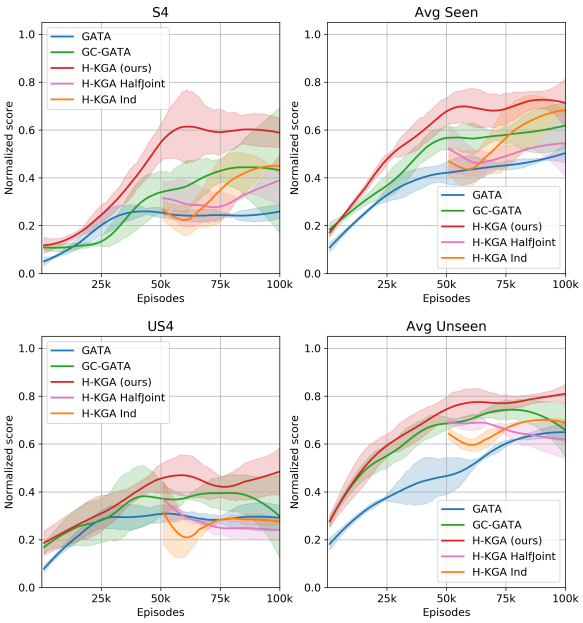

Figure 2: The models' performance on $\mathcal{D}_{\text{test}}^{\text{seen}}$ ("S4", "Avg Seen") and $\mathcal{D}_{\text{test}}^{\text{unseen}}$ ("US4", "Avg Unseen").

6 Results and discussions

6.1 Main results

Table 2 shows the testing performance at the end of training, and Fig. 2 shows the models' testing performance with respect to the training episodes. Due to space constraint, we present only results on the two most difficult levels, "S4" and "US4", as well as the average performance on $\mathcal{D}_{\text{test}}^{\text{seen}}$ and $\mathcal{D}_{\text{test}}^{\text{unseen}}$. Please refer to Appendix C for the full results. Our H-KGA outperforms baselines in both seen and unseen levels. Its advantage becomes most significant in the most complex level, "S4", which is with the most number of rooms, ingredients and required preparation steps as shown in Table 1. The performance improvement of our model can be attributed to two aspects: the goal-conditioned sub-policy and the meta-policy for adaptive goal selection. GC-GATA, which can be regarded as H-KGA without the meta-policy, also achieves improvement over GATA, demonstrating the effectiveness of goal-conditioned decision making. Compared to GC-GATA, the use of a learned meta-policy helps to further improve H-KGA.

However, we observe that joint training after pre-training the sub-policy leads to performance drop (H-KGA HalfJoint), which could be attributed to the forgetting problem in RL (Vinyals et al., 2019). Another variant, H-KGA Ind, where the pre-trained sub-policy is frozen during training the meta-policy,

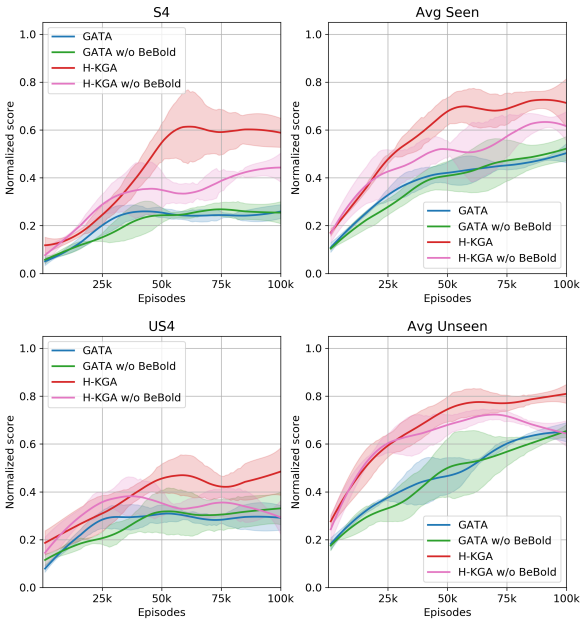

Figure 3: The models' performance with / without the BeBold reward bonus.

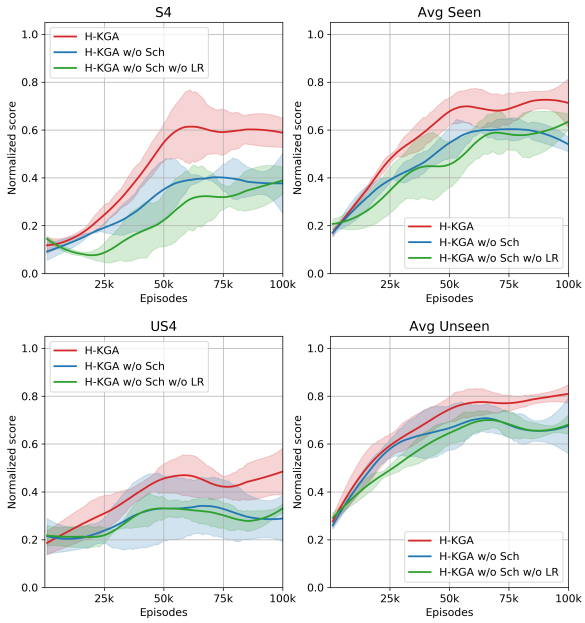

Figure 4: The models' performance with / without the multi-task learning strategies.

performs better and exceeds GC-GATA, but still worse than our H-KGA. While H-KGA Ind might still have space for performance improvement, it requires more training episodes (i.e., collecting more interaction samples), leading to low sample efficiency. Instead, our H-KGA utilizes the training data more efficiently and achieves comparable performance with fewer episodes, making it more favorable for practical applications.

We also observe that learning a good meta-policy helps in solving games from unseen levels. In "US4", where the agent has to navigate through multiple rooms to collect three ingredients, it is more important to learn a strategy to determine the collecting order. In these games, our H-KGA performs better than those without a meta-policy (GATA, GC-GATA), and those with a "not-so-good" meta-policy (H-KGA HalfJoint, H-KGA Ind).

6.2 The influence of exploration

In Sec. 4.3, we enhance the sub-policy with the BeBold reward to encourage exploration. We investigate its contribution by comparing models without such rewards. Fig. 3 shows the results. In terms of the average performance, our H-KGA is already better than GATA even without the BeBold reward ("H-KGA w/o BeBold" v.s., "GATA w/o BeBold"). However, the results on "S4" and "US4" show that sufficient exploration is essential for these difficult games, where it's hard for H-KGA without BeBold to collect over 50% (40%) of the scores in "S4" ("US4"). We also find that encouraging exploration only is not sufficient, as there is no obvious improvement for GATA, or even worse performance according to Table 2.

6.3 The influence of MTL strategies

In Sec. 4.4, we introduce two strategies to facilitate training H-KGA in the setting of multi-task learning. We then conduct ablation studies to investigate their contributions. Fig. 4 shows the results, where "Sch" denotes the **sch**eduled task sampling and "LR" denotes **l**evel-aware **r**eplay buffer. Although H-KGA can still achieve comparable average performance in both seen and unseen levels, without scheduled task sampling its performance on difficult levels, which require more training steps to collect more training samples, is limited. Similarly, training without "LR" prevents transitions of difficult levels from being added to the replay buffer, leading to low sample efficiency.

7 Conclusion

In this paper, we investigated generalization for reinforcement learning in text-based games. We introduced a two-level hierarchical framework, H-KGA, to address this problem. In the high level, a meta-policy is executed to decompose the whole game as subtasks characterized by textual goals,

and select a goal based on the knowledge graph-based observation. In the low level, a sub-policy is executed to select action conditioned on the goal. Experimental results showed that H-KGA achieved favorable performance on games with various difficulty levels. As an ongoing work, we would like to study automatic goal generation methods. We are also interested in extending our work to more complex scenarios .

Acknowledgement

This work was supported in part by ARC DP180100966. We thank Xingdi Yuan and Marc-Alexandre Côté from Microsoft Research, and anonymous reviewers for suggestions.

References

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

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

Prithviraj Ammanabrolu, William Broniec, Alex Mueller, Jeremy Paul, and Mark O Riedl. 2019. Toward automated quest generation in text-adventure games. In *Proceedings of the 4th Workshop on Computational Creativity in Language Generation*, pages 1–12.

Prithviraj Ammanabrolu and Matthew Hausknecht. 2020. Graph constrained reinforcement learning for natural language action spaces. In *International Conference on Learning Representations (ICLR)*.

Prithviraj Ammanabrolu and Mark Riedl. 2019a. 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, pages 3557–3565.

Prithviraj Ammanabrolu and Mark Riedl. 2019b. Transfer in deep reinforcement learning using knowledge graphs. In *Proceedings of the Thirteenth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-13)*, pages 1–10.

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

Prithviraj Ammanabrolu, Jack Urbanek, Margaret Li, Arthur Szlam, Tim Rocktäschel, and Jason Weston. 2021. How to motivate your dragon: Teaching goal-driven agents to speak and act in fantasy worlds. In *Proceedings of Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 807–833.

Jacob Andreas, Dan Klein, and Sergey Levine. 2017. Modular multitask reinforcement learning with policy sketches. In *International Conference on Machine Learning (ICML)*, volume 70, pages 166–175.

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

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

Dzmitry Bahdanau, Felix Hill, Jan Leike, Edward Hughes, Pushmeet Kohli, and Edward Grefenstette. 2019. Learning to understand goal specifications by modelling reward. In *International Conference on Learning Representations (ICLR)*.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. 2016. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 29, pages 1471–1479.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *International Conference on Machine Learning (ICML)*, pages 41–48.

Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. 2019. Quantifying generalization in reinforcement learning. In *International Conference on Machine Learning (ICML)*, volume 97, pages 1282–1289.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Ruo Yu Tao, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. 2018. Textworld: A learning environment for text-based games. *arXiv preprint arXiv:1806.11532*.

Peter Dayan and Geoffrey E Hinton. 1992. Feudal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 5.

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

Meng Fang, Cheng Zhou, Bei Shi, Boqing Gong, Jia Xu, and Tong Zhang. 2019a. DHER: Hindsight experience replay for dynamic goals. In *International Conference on Learning Representations (ICLR)*.

Meng Fang, Tianyi Zhou, Yali Du, Lei Han, and Zhengyou Zhang. 2019b. Curriculum-guided hindsight experience replay. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, pages 12602–12613.

Justin Fu, Anoop Korattikara, Sergey Levine, and Sergio Guadarrama. 2019. From language to goals: Inverse reinforcement learning for vision-based instruction following. In *International Conference on Learning Representations (ICLR)*.

Xiaoxiao Guo, Mo Yu, Yupeng Gao, Chuang Gan, Murray Campbell, and Shiyu Chang. 2020. Interactive fiction game playing as multi-paragraph reading comprehension with reinforcement learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7755–7765.

Hado van Hasselt, Arthur Guez, and David Silver. 2016. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 30, pages 2094–2100.

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

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

Ji He, Jianshu Chen, Xiaodong He, Jianfeng Gao, Lihong Li, Li Deng, and Mari Ostendorf. 2016. Deep reinforcement learning with a natural language action space. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1621–1630.

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

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

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

Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. 2016. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in Neural Information Processing Systems (NeurIPS)*, 29:3675–3683.

Jelena Luketina, Nantas Nardelli, Gregory Farquhar, Jakob Foerster, Jacob Andreas, Edward Grefenstette, Shimon Whiteson, and Tim Rocktäschel. 2019. A survey of reinforcement learning informed by natural language. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 6309–6317.

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

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

Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. 2018. Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, pages 3303–3313.

Karthik Narasimhan, Tejas D Kulkarni, and Regina Barzilay. 2015. Language understanding for text-based games using deep reinforcement learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1–11.

Junhyuk Oh, Satinder Singh, Honglak Lee, and Pushmeet Kohli. 2017. Zero-shot task generalization with multi-task deep reinforcement learning. In *International Conference on Machine Learning (ICML)*, volume 70, pages 2661–2670.

Baolin Peng, Xiujun Li, Lihong Li, Jianfeng Gao, Asli Celikyilmaz, Sungjin Lee, and Kam-Fai Wong. 2017. Composite task-completion dialogue policy learning via hierarchical deep reinforcement learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2231–2240.

Abdelrhman Saleh, Natasha Jaques, Asma Ghandeharioun, Judy Shen, and Rosalind Picard. 2020. Hierarchical reinforcement learning for open-domain di-

alog. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 34, pages 8741–8748.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2015. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.

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

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

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

Tianmin Shu, Caiming Xiong, and Richard Socher. 2018. Hierarchical and interpretable skill acquisition in multi-task reinforcement learning. In *International Conference on Learning Representations (ICLR)*.

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. 2016. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489.

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

Richard S Sutton, Doina Precup, and Satinder Singh. 1999. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211.

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. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 673–683.

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

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. 2017. FeUdal networks for hierarchical reinforcement learning. In *International Conference on Machine Learning (ICML)*, volume 70, pages 3540–3549.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. 2019. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354.

Yunqiu Xu, Ling Chen, Meng Fang, Yang Wang, and Chengqi Zhang. 2020a. Deep reinforcement learning with transformers for text adventure games. In *IEEE Conference on Games (CoG)*, pages 65–72.

Yunqiu Xu, Meng Fang, Ling Chen, Yali Du, Joey Tianyi Zhou, and Chengqi Zhang. 2020b. Deep reinforcement learning with stacked hierarchical attention for text-based games. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 16495–16507.

Shunyu Yao, Rohan Rao, Matthew Hausknecht, and Karthik Narasimhan. 2020. Keep CALM and explore: Language models for action generation in text-based games. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8736–8754.

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

Xusen Yin, Ralph Weischedel, and Jonathan May. 2020. Learning to generalize for sequential decision making. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing: Findings (EMNLP-Findings)*, pages 3046–3063.

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

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 *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, pages 3562–3573.

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