Language-based General Action Template for Reinforcement Learning Agents

Ryosuke Kohita
IBM Research
kohi@ibm.com

Akifumi Wachi
IBM Research
akifumi.watchi@ibm.com

Daiki Kimura
IBM Research
daiki@jp.ibm.com

Subhajit Chaudhury
IBM Research
subhajit@jp.ibm.com

Michiaki Tatsubori
IBM Research
mich@jp.ibm.com

Asim Munawar
IBM Resarch
asim@ibm.com

Abstract

Prior knowledge plays a critical role in decision-making, and humans preserve such knowledge in the form of natural language (NL). To emulate real-world decision-making, artificial agents should incorporate such generic knowledge into their decision-making framework through NL. However, since policy learning with NL-based action representation is intractable due to NL's combinatorial complexity, previous studies have limited agents' expressive power to only a specific environment, which sacrificed the generalization ability to other environments. This paper proposes a new environment-agnostic action framework, the language-based general action template (L-GAT). We design action templates on the basis of general semantic schemes (FrameNet, VerbNet, and WordNet), facilitating the agent in finding a plausible action in a given state by using prior knowledge while covering broader types of actions in a general manner. Our experiment using 18 text-based games showed that our proposed L-GAT agent which uses the same actions across games, achieved a performance competitive with agents that rely on game-specific actions. We have published the code at https://github.com/kohilin/lgat.

1 Introduction

The incorporation of natural language processing (NLP) and reinforcement learning (RL) is an important research field for using knowledge, represented in the form of language, in the decision-making of artificial agents (Luketina et al., 2019). One critical topic is the capability to describe an agent's actions with natural language (NL) (Narasimhan et al., 2015; Yuan et al., 2019). An agent with such a capability can estimate the plausibility of actions on the basis of prior knowledge accessible through

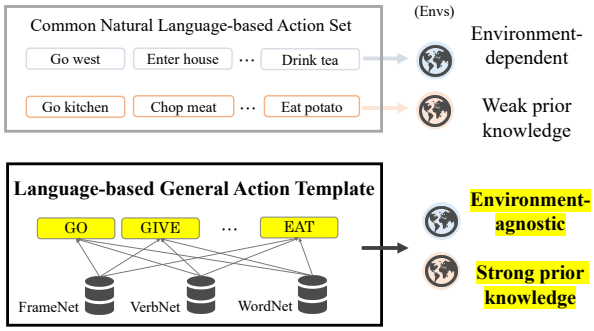

Figure 1: Common action templates and L-GAT.

language (Narasimhan et al., 2018). Suppose that an agent receives a request, "*give me some water*"; a common sense idea like "*water exists in the kitchen*" will definitely help the agent determine the right direction to go in. If actions are represented in language, for example, GO TO KITCHEN, we can straightforwardly connect knowledge to actions by referring to language resources (Fulda et al., 2017). NL is useful for accessing knowledge to achieve plausible decision-making, and this language capability would be fundamental in developing intelligent agents.

However, NL is complicated for current RL agents to acquire due to its high expressive power (Hausknecht et al., 2020). The rich vocabulary and complex grammar of NL result in a huge action space that is intractable for existing RL algorithms. Although we can train an agent by restricting the expressive power to a specific environment (Narasimhan et al., 2015; He et al., 2016), such constraints sacrifice the inherent advantage of using NL as the action representation for domain-independent learning. Our objective is to make better trade-offs between expressive power and the learnability of NL-based RL.

Interactive fiction (IF) games serve as a practical testbed for NL-based RL, where an agent and environment communicate with each other through

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2125–2139

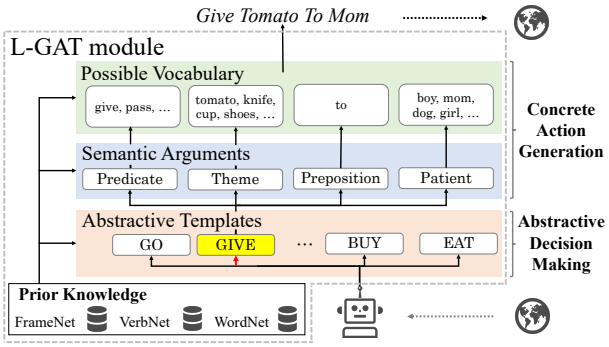

Figure 2: Action generation flow with L-GAT.

textual information (Côté et al., 2019). The agent needs to understand a textual state and generate an appropriate NL action command. In the experiments of previous studies, various constraints on the action space have been used for the convergence of learning, such as simple grammar and vocabulary (Narasimhan et al., 2015), effective ground-truth actions for a state (He et al., 2016), and game-specific templates (Hausknecht et al., 2020). These constraints are problematic, especially when applying the same algorithm to other environments. There are several studies that address this by implementing heuristic rules (Hausknecht et al., 2019) or training an action generator with observation-action pairs of human-play (Yao et al., 2020), but they still have a non-trivial bias toward IF games.

In terms of prior knowledge, word embeddings and language models have shown success at reducing the action space (Fulda et al., 2017; Yao et al., 2020). Although such continuous representations have achieved notable performances for broader NLP tasks, they are basically trained with word co-occurrences. We cannot flexibly manipulate the knowledge contained in these continuous representations because it is difficult to selectively encode the desired knowledge into them (Zhou et al., 2020). To make the most of prior knowledge expressed in NL, we should take advantage of other linguistic resources as well that provide fine-grained information more explicitly, such as the hierarchical structure of words (Miller, 1998), the semantics of a sentence (Fillmore et al., 2003), and common sense (Speer et al., 2017).

Thus, we propose a new environment-agnostic action framework on the basis of general semantic schemes, the language-based general action template (L-GAT). Figure 1 shows the overall architecture compared with the environment-dependent action templates commonly used, and Figure 2 illustrates the flow of generating an action with L-GAT. The agent with L-GAT first determines "*what to do*" at an abstractive level with generally defined action templates and then specifies "*how to do it*" by generating a concrete action on the basis of connected prior knowledge.

Our contributions with L-GAT are three-fold. First, we propose an environment-agnostic action framework based on general semantic schemes such as FrameNet (Fillmore et al., 2003), VerbNet (Schuler, 2006), and WordNet (Miller, 1998). Second, we develop a hierarchical action-generation algorithm in which the agent performs abstractive decision-making and then generates a concrete action command. Third, we introduce a method for dynamically reducing the action space with static knowledge from multiple external resources and contextual information from a state. We have published the code with which future studies can easily use L-GAT as an action generation module.

2 Related Work

The action space problem of NL-based actions has been addressed in previous work. LSTM-DQN (Narasimhan et al., 2015) enables an agent to learn a policy in a simple verb-object format in synthetic environments. DRRN (He et al., 2016) is a ranking-based method in which the agent selects an action from among admissible actions given by a game engine. Game-specific templates were proposed for Jericho (Hausknecht et al., 2020). The agent selects one of the templates and fills in the gaps in the chosen template. TDQN (Hausknecht et al., 2020) and KGA2C (Ammanabrolu and Hausknecht, 2020) agents have used these templates and succeeded at learning a policy in various games. NAIL (Hausknecht et al., 2019) solves IF games with action generation heuristics.

Language resources have also been used for reducing the action space. Fulda et al. (2017) extracted the affordances of words with word embeddings and restricted objects that can be taken for a specific verb. A language model is leveraged to filter non-plausible word combinations (Kostka et al., 2017). Recently, Yao et al. (2020) proposed the contextual action language model (CALM), which trains a language model with human-play

transcriptions and uses it as an action candidate generator. By combining it with a ranking-based method, CALM showed significant performance even without a ground-truth for admissible actions.

3 Problem Setting

In this section, we formally define our problem setting, RL agents with NL-based actions. An NL-based action a, such as GO TO KITCHEN and GIVE SOME WATER TO JOHN, consists of N words: $a = \{w_1, w_2, \ldots, w_N\}$, $w_i \in \mathcal{V}$, where each w_i is a word and $\mathcal{V}$ is vocabulary. Given a state s, we represent an optimal NL-based action in the state with $a^*_{(s)}$ and the words composing it with $w^*_{(i,s)}$, respectively. Following the previous studies, each $w \in \mathcal{V}$ is estimated with an independent Q-function as $Q_{(i)}(s, w)$, where i is the position of a word, and $\mathcal{Q}(s, a)$ is defined as $\Sigma_i Q_{(i)}(s, w)$. A policy π is evaluated with a cumulative reward as:

$$V^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \,\middle|\, s_0 = s, a_0 = a\right],$$

where R is a reward function for a state-action pair, and γ is a discount factor. Then, the Q-functions $\mathcal{Q}^*, Q^*$ corresponding to the optimal policy are learned by minimizing the following loss:

$$\mathcal{L} = (R(s, a) + \gamma \bar{\mathcal{Q}}^*(s', a') - \mathcal{Q}^*(s, a))^2,$$

where $\bar{\mathcal{Q}}^*(s', a') = \Sigma_i \max_{w'} \bar{Q}^*_{(i)}(s', w')$ with the next state s', the next word w', and the target Q-function $\bar{Q}$. Then, we obtain each word of the optimal action as: $w^*_{(i,s)} = \arg\max_w Q^*_{(i)}(s, w)$. However, this optimization could be intractable depending on N and the size of $\mathcal{V}$ due to the exponential complexity $\mathcal{O}(|\mathcal{V}|^N)$.

Hence, we consider restricting the vocabulary space for each position i for the action and state s to make this optimization problem tractable. Specifically, we want to find a subset $\mathcal{V}_{(i,s)}$ of $\mathcal{V}$ for each (i, s) such that $|\mathcal{V}_{(i,s)}|$ is much smaller than $|\mathcal{V}|$. Also, $\mathcal{V}_{(i,s)}$ must include the optimal word $w^*_{(i,s)}$ for the original optimization problem because replacing $\mathcal{V}$ with $\mathcal{V}_{(i,s)}$ should not change the optimal solution. Assuming that we have $\mathcal{V}_{(i,s)} \subset \mathcal{V}$, the computational complexity decreases from $\mathcal{O}(|\mathcal{V}|^N)$ to $\mathcal{O}(\prod_{i=1}^N |\mathcal{V}_i|)$, where $\mathcal{V}_i$ is one of the largest $\mathcal{V}_{(i,s)}$ for all s.

To define $\mathcal{V}_{(i,s)}$, we use knowledge $\mathcal{K}$:

$$\mathcal{V}_{(i,s)} = \varphi(i, s, \mathcal{K}, \mathcal{V}),$$

where φ is a set-valued function. Then, we need to determine the function φ that minimizes $|\mathcal{V}_{(i,s)}|$ but keeps the i-th word of an optimal NL-based action in $\mathcal{V}_{(i,s)}$:

$$\min |\mathcal{V}_{(i,s)}|, \text{ s.t. } w^*_{(i,s)} \in \mathcal{V}_{(i,s)}.$$

Note that, although the above formulation assumes that $\mathcal{V}_{(i,s)}$ always contains the optimal words $w^*_{(i,s)}$, it is practically impossible because the optimal words are unknown a priori and estimated through the learning process. Therefore, we need to approximate φ so that the estimated $\mathcal{V}_{(i,s)}$ is likely to contain $w^*_{(i,s)}$ as much as possible. We introduce our implementation of the approximation in Section 4.4.

4 Method

We now introduce our proposed method; the language-based general action template (L-GAT). L-GAT is a framework of NL-based actions for environment-agnostic RL agents. In L-GAT, we define actions on the basis of general semantic schemes, such as FrameNet (Fillmore et al., 2003), VerbNet (Schuler, 2006), and WordNet (Miller, 1998), which enables an agent to use connected knowledge to reduce the action space.

In this section, we first provide an overview of L-GATand then give details on its action command, definition, and generation algorithm.

4.1 Overview

L-GAT adopts hierarchical modeling for action generation to handle the vast space of NL-based action. Specifically, an agent with L-GAT first determines "*what to do*" at an abstractive level (e.g., decide to **give**) and then next determines "*how to do it*" at a concrete level (e.g., decide to give **tomato** to **mom**) as shown in Figure 2. This two-step strategy is intuitive and natural as a general procedure for decision-making (Lazaridou et al., 2020). From the viewpoint of RL, we can reduce the action space because the number of abstractive actions is much smaller than that of all possible words in the whole vocabulary. Also, the chosen abstractive action further restricts the words available for generating a complete action (e.g., we cannot eat a table)

To use such hierarchical modeling, L-GAT defines the **Abstractive Action Template** with a hierarchical structure consisting of three components: **Frame**, **Role**, and **Lexicon**. Figure 3 illustrates the hierarchical structure of

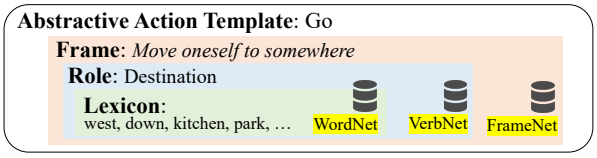

Figure 3: Abstractive action template GO and its hierarchical components; Frame, Role, and Lexicon.

an abstractive action template, GO. The frame defines the semantics of GO (e.g., *move oneself to somewhere*), the role represents a conceptual argument required to perform the action (e.g., `Destination`), and the lexicon represents concrete words that can be used for one of the roles in the frame (e.g., *west*, *down*, and *kitchen*). Using the generation algorithm, the agent selects an appropriate word from the lexicon for each frame's role in the chosen abstractive action template.

FrameNet and VerbNet inspired the hierarchical structure of L-GAT. Both resources provide semantic schemes that describe our daily behaviors conceptually. For example, similar to the GO template, FrameNet has a relevant scheme, "self-motion", with required arguments such as Goal and Source. Also, we connect the lexicon of L-GAT to WordNet to represent the hierarchical relationships of words. The word hierarchy enables agents to generalize candidate words (e.g., hyponyms of "*direction*," such as "*west*," can be used for the GO template) and interact with diverse environments. Thus, by designing actions on the basis of these general semantic schemes, we can naturally make L-GAT environment-agnostic and familiar with general prior knowledge.

Note that the templates of L-GAT are easily configurable depending on the environment. Such customizability is practically essential to applying the scheme to diverse domains, which is costly or difficult for an environment-dependent action set or language model-based generator.

4.2 Action Command

L-GAT generates an action command in a fixed format as: $a = v + n_1 + p + n_2$, with at most four slots for a verb (v), two nouns (n_1, n_2), and a preposition (p). An action command is generated by the action templates explained in the next section, and n_1, n_2, and p are not necessarily used depending on the template. To prevent the exponential growth of the action space, we decided to add these hard constraints to a generable format. However, we consider the aforementioned format to be able to cover most of the primary actions demanded by agents, such as *go kitchen* and *put cup on table*.

4.3 Action Definition

This section introduces the abstractive action template and its components. In this paper, we manually defined 41 abstractive action templates for L-GAT (see Appendix A). For the construction of each template, we chose relevant semantic schemes from FrameNet and VerbNet, and then aggregated them in a compatible form with the actions of RL agents. Table 1 shows three templates as references, and we will refer to them throughout this section.

Abstractive Action Template. A template T is defined as a tuple $T = \langle f, r_{(v)}, r_{(n_1)}, r_{(p)}, r_{(n_2)} \rangle$ where f is the frame, and $r_{(v|n_1|p|n_2)}$ are the roles for each slot. The frame defines the template's semantics. The roles represent required arguments for performing an action. Each word in an action command is an instance of its corresponding role and selected from the lexicon explained later.

Frame. Each frame has connections to semantically relevant entities in FrameNet (called *Semantic Frames*), and ones in VerbNet optionally (called *Verb Classes*). For example, the GIVE template is related to the semantic frame *Giving* and the verb class *give-13.1*. We use FrameNet as the basis because the granularity of the descriptions is more suitable for our purpose than VerbNet.

Role. For the definition of our roles, we borrow VerbNet's *Thematic Roles*, which refer to the semantic relationship between a predicate and its argument. VerbNet defines 30 thematic roles in total, and L-GAT uses 15 of them by taking into account their frequency and meaning. FrameNet also has a similar concept called *Frame Elements*; however, it is too fine-grained for RL agents' actions. Therefore, we use the thematic roles of VerbNet as the basis and annotate the related frame elements as additional information. We also define three special roles: `Predicate`, `Preposition`, and `Null`. `Predicate` is used only for the verb slot and references to verb nodes in WordNet. We selected these nodes on the basis of the representative verbs given by the connected frame entities of FrameNet and VerbNet (e.g., we chose "go.v.02" and "enter.v.01" for the

	f	$r_{(v)}$	$r_{(n_1)}$	$r_{(p)}$	$r_{(n_2)}$
L-GAT	GO	`Predicate`	`Destination`	`Null`	`Null`
FrameNet	Self-motion	-	Goal, Place, Area, Path	-	-
VerbNet	-	-	Destination	-	-
WordNet	-	go.v.02, enter.v.01	location.n.01, ...	-	-
L-GAT	GIVE	`Predicate`	`Theme`	`Preposition`	`Recipient`
FrameNet	Giving	-	Theme	-	Recipient
VerbNet	give-13.1	-	Theme	-	Recipient
WordNet	-	give.v.01	object.n.01	(to)	people.n.01, person.n.01
L-GAT	EAT	`Predicate`	`Patient`	`Null`	`Null`
FrameNet	Ingestion	-	Food	-	-
VerbNet	eat-39.1-1	-	Patient	-	-
WordNet	-	eat.v.01	food.n.01	-	-

Table 1: Examples of abstractive action templates in L-GAT with connections to FrameNet, VerbNet, and WordNet.

`Predicate` of the GO template on the basis of representative verbs given by the semantic frame "self-motion"). `Preposition` has prepositions available that often accompany verbs in a frame, and we manually defined these prepositions. `Null` means not using the slot for action generation. For example, the GO template uses only v and n_1.

Lexicon. The simplest way of defining the lexicon is to list all possible words manually or statistically; however, such a procedure would be non-scalable or hard to configure. Therefore, we decided to annotate nodes in WordNet as the representation. We selected general nodes such as "location.n.01" for the `Destination` of the GO template and "food.n.01" for the `Patient` of the EAT template. The hierarchical word relations in WordNet enables L-GAT to structurally determine candidate words such as by choosing the hyponyms of annotated nodes.

4.4 Action Generation

We propose three techniques for action generation with L-GAT: **Hierarchical Prediction**, **Word Masking**, and **Template Masking**. An agent with L-GAT generates an action as follows. The agent determines the abstractive action template and then selects a word for each slot from among candidates (Hierarchical Prediction). The candidates for each slot are filtered by using information from the state and the chosen template (Word Masking). By precomputing Word Masking for all templates in advance of the template selection, L-GAT can provide possible templates for a state by excluding templates that cannot generate any action command (Template Masking). The algorithm is described in Algorithm 1 and Figure 4. L-GAT has five Q-functions; one for estimating the template $Q(s, T)$, and the others for estimating words $Q_{(\cdot)}(s, w)$ where $(\cdot)$ denotes one of the slots v, n_1, p, or n_2.

Algorithm 1 Action Generation with L-GAT.

1: Input: s, Output a
2: $\mathcal{V}_{(\cdot,*,s)} \leftarrow$ compute the restricted vocabulary for all templates
3: $\mathcal{T} \leftarrow$ exclude templates with empty $\mathcal{V}_{(\cdot,T,s)}$
4: $T \leftarrow$ select a template from $\mathcal{T}$
5: $v, n_1, p, n_2 \leftarrow$ select words from $\mathcal{V}_{(\cdot,T,s)}$
6: $a \leftarrow$ decode with v, n_1, p, n_2

4.4.1 Hierarchical Prediction

L-GAT first selects a template T with $Q(s, T)$ and then generates words for each slot as: $w_{(\cdot)} = \arg\max_{\hat{w}} Q_{(\cdot)}(s, \hat{w}), \hat{w} \in \mathcal{V}_{(\cdot,T,s)}$, where $\mathcal{V}_{(\cdot,T,s)}$ is vocabulary that is restricted for the target slot of the template T in the state s. L-GAT computes $\mathcal{V}_{(\cdot,T,s)}$ as: $\mathcal{V}_{(\cdot,T,s)} = \{w \mid m(\cdot, s, T, w) = 1, w \in \mathcal{V}\}$, where m is a *masker function* that returns 1 for a generable word in the target slot by considering s and T. The masker function is our approximation of φ explained in Section 3, and we introduce the details in the next section.

4.4.2 Word Masking

We included five sub-masker functions; Role masker, Language Model (LM) masker, Part-of-Speech (PoS) masker, Stopword masker, and Observation masker. Each of them returns 1 for generable words. Then, we define $m(\cdot, s, T, w) = 1$ only when all the sub-maskers return 1. Next we give the definitions of the five sub-maskers.

Role masker. The Role masker filters words that are not semantically compatible with a given role

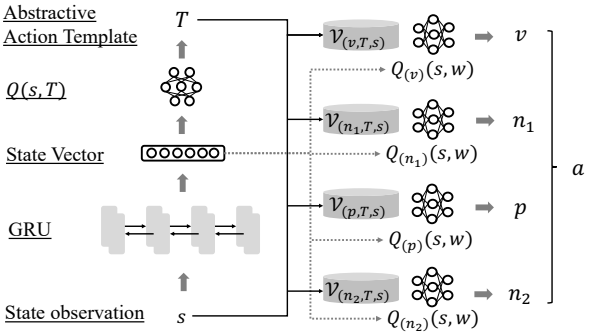

Figure 4: L-GAT model without Template Masking.

Limitations	Methods				
	TD	KG	NA	CA	LG
Game-specific vocab	y	y	y	n	y
Game-specific template	y	y	n	n	n
Admissible actions	y	y	n	n	n
Human play data	n	n	n	y	n
Tuned Rules for IFGs	n	n	y	n	n
Exclude no-effect acts	y	y	n	y	n

Table 2: Limitations in action generation. Method names come from their first two characters.

by referring to the annotated WordNet nodes as mentioned in §4.3. For the noun slots, it enables hyponym words of the reference nodes to be produced. For example, for the n_1 slot of the EAT template, it returns 1 for hyponym words of the *food.n.01* node such as *water* and *tomato* (see Table 1). For the v slot (i.e., `Predicate`), it returns 1 for lemmas of the annotated nodes themselves such as *go.v.01* = *go* and *enter.v.01* = *enter*. All words have 1 in the p slot, and no words have 1 in any `Null` slot.

LM masker. The LM masker filters contextually irrelevant candidates with an LM. The LM takes an observation suffixed with a verb of the template as the input (an example is given in Appendix B), and it predicts the following word. On the basis of the estimated probability of next words, the LM masker returns 1 for the top-k words. At the v and p slots, all words have 1. We used a pre-trained GPT-2 (Radford et al., 2019) and set $k = 50$.

PoS, Stopword, and Observation maskers. The PoS masker filters words in which a PoS is not matched with the expected one. Specifically, it returns 1 for verbs in the v slot, for nouns in the n_1 and n_2 slots, and for prepositions in the p slot. The word-PoS mapping follows WordNet in terms of verbs and nouns, and we prepared a fixed list of prepositions. The Stopword masker prohibits the generation of stopwords such as determiners (e.g., *a*, *the*) and pronouns (e.g., *he*, *them*). We prepared a fixed list, and the masker returns 1 for words that are not on the list. The Observation masker filters unseen words from observation. It returns 1 for words that appeared in the last observation.

4.4.3 Template Masking

L-GAT also filters unavailable templates in a state with the $\mathcal{V}_{(\cdot,T,s)}$ of each template. $\mathcal{V}_{(\cdot,T,s)}$ is empty when one of the sub-maskers denies the use of a word, and this occurs for all words in $\mathcal{V}$. An empty $\mathcal{V}_{(\cdot,T,s)}$ means that no appropriate objects exist for performing the action (e.g., we cannot eat anything if no eatable objects exist.). Thus, L-GAT selects a template as: $T = \arg\max_{T \in \mathcal{T}} Q(s,T)$, where $\mathcal{T}$ is a set of templates in which $\mathcal{V}_{(\cdot,T,s)}$ is not empty for all slots except for NULL slots.

5 Experiment

In our experiment, we measured performance in IF games in Jericho (Hausknecht et al., 2020). We compared our L-GAT agent with agents who rely on environment-dependent actions. We also performed detailed analysis such as on action coverage and action space reduction and masker ablation study on L-GAT. Furthermore, to evaluate the generalization ability of L-GAT, we also conducted an extended experiment in which a single agent solved all games.

5.1 Settings

Game environment. We selected 18 games in Jericho on the basis of the performance in previous studies. We did not select any game that was too hard for most of the existing agents to learn.

Baselines. There were four existing agents tested in Jericho; TDQN (Hausknecht et al., 2020), KGA2C (Ammanabrolu and Hausknecht, 2020), NAIL (Hausknecht et al., 2019), and CALM (Yao et al., 2020). The limitations related to their action generation are summarized in Table 2 (see also §2).

In the experiments with TDQN, KGA2C, and CALM, these methods preselected steps by excluding no-effect actions that do not change the world state for faster and stable learning. This limitation is not appropriate for testing general action sets such as L-GAT because the game engine explicitly restricts the action space.

Games	Max Score	TDQN (w/ step preselection)	KGA2C (w/ step preselection)	CALM (w/ step preselection)	NAIL (w/o step preselection)	TDQN+ (w/o step preselection)	L-GAT (w/o step preselection)	Cover Ratio (#)	
balances	51	4.8	10.0	9.1	**10.0**	9.8	8.8	53%	(89)
deephome	300	1.0	1.0	1.0	13.13	13.2	**14.9**	87%	(100)
detective	360	169.0	207.9	289.7	136.9	**270.9**	214.1	100%	(46)
enchanter	400	8.6	12.1	19.1	0.0	**22.9**	0.0	73%	(105)
gold	100	4.1	-	-	**3.0**	0.0	0.0	86%	(331)
inhumane	90	0.7	3.0	25.7	**0.6**	0.0	0.0	95%	(84)
jewel	90	0.0	1.8	0.3	**1.6**	0.6	0.0	74%	(81)
library	30	6.3	14.3	9.0	0.9	4.1	**7.6**	69%	(29)
ludicorp	150	6.0	17.8	10.1	8.4	**11.9**	6.1	97%	(152)
omniquest	50	16.8	16.8	6.9	**5.6**	4.4	5.0	88%	(75)
reverb	50	0.3	-	-	0.0	0.0	**1.0**	82%	(65)
snacktime	50	9.7	0.0	19.4	0.0	**10.0**	0.0	88%	(26)
spellbrkr	600	18.7	21.3	40.0	**40.0**	37.6	39.4	57%	(109)
spirit	250	0.6	1.3	1.4	1.0	1.5	**1.7**	89%	(209)
temple	35	7.9	7.6	0.0	**7.3**	2.6	5.0	94%	(80)
zork1	350	9.9	34.0	30.4	10.3	4.7	**17.1**	95%	(203)
zork3	7	0.0	0.1	0.5	**1.8**	0.3	0.4	92%	(192)
ztuu	100	4.9	9.2	3.7	0.0	3.6	**4.9**	62%	(78)
Norm Avg.	-	9.8%	14.1%	15.5%	8.0%	9.9%	8.9%	82%	

Table 3: Average scores of last 100 episodes. Those of TDQN, KGA2C, NAIL, and CALM come from their respective papers. TDQN, KGA2C, and CALM cannot be fairly compared with L-GAT because of step preselection by excluding no-effect actions. TDQN+ and L-GAT are our implementations containing all steps regardless of action effectivity. Cover ratio shows proportion of generable walkthrough actions with L-GAT, and number in parentheses is total number of effective walkthrough actions.

To focus on the comparison of environment-dependent and agnostic action templates, we re-implemented the TDQN algorithm in our code base (denoted as TDQN+), in which all steps were counted regardless of action effectivity. Also, we implemented TDQN+ without the admissible action limitation; the difference in the above limitations between TDQN+ and L-GAT is their template types only (i.e., game-specific or general templates). Thus, we mainly compared NAIL, TDQN+, and L-GAT. Although a comparison with the original scores of TDQN, KGA2C, and CALM would not be fair, we also add them as references for completeness.

Note that the vocabulary limitation is practically needed because any action containing unknown words for a game engine cannot be accepted even though the action command is semantically correct. CALM overcame this vocabulary issue by learning the word distribution of action commands used in IF games from human-play transcriptions.

Model and training details. Following the works of Hausknecht et al. (2020); Yao et al. (2020), we used a bidirectional GRU as our observation encoder, and the observation string was augmented by combining the observations returned by the "look" and "inventory" commands. All Q-functions were implemented with a multi-layered perceptron with the same hyperparameters. In an episode, we limited the number of steps to 100 at most and ran agents with ten environments in parallel. In addition to the rewards given by the games, we gave an exploration bonus of 1 when an agent found an unseen state in the episode (Yuan et al., 2019). We put the hyperparameter details in Appendix C. We trained three agents with different random seeds and used their average score for the evaluation.

5.2 Results

We report the results of the game performance, action coverage by L-GAT, action space reduction by L-GAT, and an ablation study on Word Masking. Finally, we introduce the results of a single agent that solved multiple games.

Game performance. Following the previous studies, we calculated the average score of the last 100 episodes, and the results are shown in Table 3. The average normalized score (raw score divided by the maximum score) was 8.0% for NAIL, 9.9% for TDQN+, and 8.9% for L-GAT. Even though the action templates of L-GAT were designed in a general manner, our agent achieved reasonable performance across games and outperformed NAIL and TDQN+ in six games.

NAIL and L-GAT performed poorly in enchanter and snacktime compared with TDQN+. In these

games, a number of critical actions for advancing the story were missing because their generable actions are defined outside of a specific game. For example, artificial words like spells (e.g., *frotz*, *gnusto*) appear in enchanter, and they are intractable with L-GAT, which uses only common language. As an additional investigation, we added a "SPELL" action to L-GAT for producing these spell words and re-trained an agent with the enchanter game; the score of L-GAT then increased to 20.0, which was close to the score of TDQN+ (22.9).

The scores of KGA2C (14.1%) and CALM (15.5%) were significantly higher. This suggests that their techniques, such as the dynamic state graph encoding and the domain adaptation of LM's outputs, are promising, and we will investigate their importance for our framework as future work.

Walkthrough coverage. Jericho provides an action trajectory called a *walkthrough* that solves a game. We assessed how much L-GAT covered walkthrough actions. The detailed procedure of this assessment is explained in Appendix C. The right-most column in Table 3 shows the cover ratio of the walkthrough actions for L-GAT. The average coverage across games was 82%, and around 90% or more of the actions were covered in half of the games. A lower coverage ratio ($< 70\%$) was observed in balances (53%), spellbrkr (57%), ztuu (62%), and library (69%). In their walkthrough actions, we frequently found artificial words (e.g., *rezrov*), named entities such as person's name (e.g., *give card to Alan*), and modifiers for objects (e.g., *wear fur suit*, *take ticket 2306*). Although the first case might be out-of-scope of L-GAT, the second and third are critical because they must appear in an interaction with real-world environments.

Action space reduction. We evaluated how much the action space was reduced in L-GAT by applying Word Masking (4.4.2) and Template Masking (4.4.3). For each slot in each available template (v, n_1, p, and n_2), we counted the number of words that were accepted by these techniques. Compared with the size of the whole vocabulary, the number of words was significantly reduced by 94% for v (24.8 words on average after reduction), 82% for n_1 (117.9 words), 99% for p (7.7 words), and 59% for n_2 (292.2 words).

Masker ablation. Figure 5 shows the learning curves of the L-GAT agents without one of the sub-maskers for Word Masking in zork1. A significant effect was observed with the Role masker. Specifically, the agent without the Role masker took more episodes to find the scored trajectory and degraded the overall performance. The PoS and Stopword maskers were not critical, which might be because their restrictions are already covered by the Role masker. The LM and Observation maskers had non-trivial effects. Since the Role masker enables relatively broader words in noun slots (i.e., hyponyms of a general word), contextual restrictions given by the LM and Observation maskers helped the agent identify available objects.

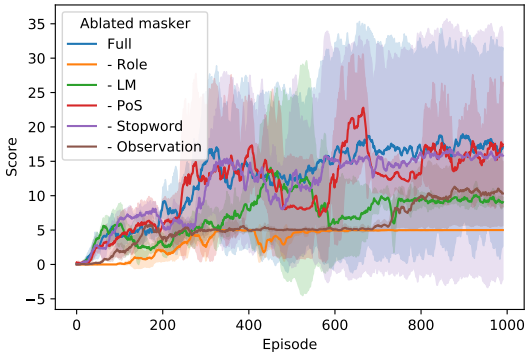

Figure 5: Masker ablation of L-GAT in zork1.

Game	Multi-task		Zero-shot	
	TDQN+	L-GAT	TDQN+	L-GAT
balances	0.0	1.5	0.0	0.8
deephome	1.0	1.1	1.0	1.1
detective	39.0	42.7	16.7	25.2
enchanter	0.0	0.0	0.0	1.4
gold	0.0	0.0	0.0	0.0
inhumane	0.0	0.0	0.0	0.0
jewel	0.0	0.0	0.0	0.0
library	0.3	0.0	0.0	0.0
ludicorp	4.2	4.1	2.8	2.3
omniquest	1.6	2.1	0.0	0.5
reverb	0.0	0.2	0.0	0.0
snacktime	0.1	0	0.0	0.0
spellbrkr	6.1	14.7	2.0	8.2
spirit	0.7	0.2	0.0	0.1
temple	0.0	0.0	0.0	0.0
zork1	0.0	0.4	0.0	0.0
zork3	0	0.0	0.0	0.0
ztuu	0	2.3	0.0	0.6
Norm avg.	1.1%	1.5%	0.4%	0.7%

Table 4: Average scores of last 100 episodes in game-independent agent experiments.

Game-independent agent To investigate the generalization ability of L-GAT, we conducted a multi-task experiment where a single agent solved

all the games. We prepared two types of agents: 1) a *multi-task agent*, who learns with all 18 games and solves each game, and 2) a *Zero-shot agent*, who learns with 17 games and solves the remaining unseen game. Note that L-GAT can be naturally applicable to this experimental setting. In terms of the implementation of TDQN+, we accumulated all the unique templates used in training games as available templates. Table 4 shows the game scores. Both TDQN+ and L-GAT largely decreased in score compared with Table 3, in which we trained and tested a specific agent for each game. Also, as expected, the zero-shot situation was more difficult. Whereas L-GAT outperformed TDQN+, the scores were far from achieving the goal. It was still challenging to obtain an environment-agnostic agent even though the action set itself is defined in a general manner.

6 Limitations and Future Work

Although L-GAT showed a better performance in our experiment, there is still room for improvement to enhance its language capability. We here discuss L-GAT's limitations and future improvements.

Experiments in environments other than text-based games. One of the advantages of L-GAT is its action generality. Jericho is a reasonable testbed for L-GAT because of the diverse games and strong NL interpreter. However, the performance in other environments such as dialogue systems (Dinan et al., 2019) and 3D video games (Gordon et al., 2018) will also be critical metrics because objectives for IF-game agents are not necessarily compatible with ones for real-world agents. Testing with real-world like environments will lead to insights for improving L-GAT.

Enhancement of expressive power. In our experiment, L-GAT had a high action coverage in the IF games. However, the format of generable actions is fixed, and we observed several critical disabilities with generating frequently used expressions. For example, L-GAT cannot cope with multi-word expressions (e.g., *turn on*, *New York*) and modified nouns (e.g., *red cup*, *dog in the room*). Note that we can make them generable without adding extra slots, such as by inserting multi-word expressions in the vocabulary and integrating a reference resolution module. A technique that increases the expressive power but keeps or decreases the action space would be a desirable enhancement.

Zero-shot learning in L-GAT The general-purpose property of L-GAT can be seen as a zero-shot learning problem. Recently, Jain et al. 2020 proposed a new experiment to assess the adaptability of an RL agent to unseen states such as by using a new tool with the knowledge of known tools. L-GAT potentially can work for such a zero-shot situation. For example, let us assume that an L-GAT agent has a learned policy for attacking a monster with a hammer, and he then encounters a monster, but only a sword is available. Even if the agent did not know how to or even use a sword at all, he may be able to use it because L-GAT has knowledge on attacks with "*weapon*" (i.e., the `Instrument` role of the ATTACK template has a weapon node in WordNet, and "*sword*" is a hyponym of "*weapon*"; see Table 1 in Appendix A). Investigation into the adaptability to unseen states would be a promising analysis for L-GAT.

Connection to wide NLP resources and tools. Although we developed L-GAT on top of general semantic schemes such as FrameNet, VerbNet, and WordNet, the required information for efficient decision-making cannot be covered with only those resources. Intelligent agents must comprehend common sense, causality, and world-knowledge (Luketina et al., 2019). Research on NLP has created various resources such as for retrieving information from Wikipedia (Chen et al., 2017), inferring the next action from a current state (Zellers et al., 2018), and using a common-sense database (Speer et al., 2017). Integration of such resources into L-GAT is critical future work, and the three general semantic schemes would be helpful for bridging different knowledge bases.

7 Conclusion

We proposed L-GAT as an environment-agnostic language-based action framework for RL agents. We designed L-GAT so that an agent can have a higher language capability to generate its own actions while keeping the action space tractable by using prior knowledge. Our experiment with multiple IF games showed that the L-GAT agent competitively performed against agents with game-specific actions. We discussed the current limitations of L-GAT and its future improvements.

Ethical Consideration

Our research is on defining a new action framework for RL agents. Our proposed method, L-GAT, is for generating actions of RL agents, which is unlikely to produce ethically problematic sentences such as those in hate speech. Also, we can control the vocabulary so that an agent with L-GAT does not produce such problematic sentences. Therefore, we consider the ethical risks of L-GAT to be low and controllable.

References

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

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

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. 2019. Textworld: A learning environment for text-based games. *arXiv preprint arXiv:1806.11532*.

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. 2019. Wizard of wikipedia: Knowledge-powered conversational agents. In *International Conference on Learning Representations*.

Charles J. Fillmore, Christopher R. Johnson, and Miriam R.L. Petruck. 2003. Background to Framenet. *International Journal of Lexicography*, 16(3):235–250.

Nancy Fulda, Daniel Ricks, Ben Murdoch, and David Wingate. 2017. What can you do with a rock? affordance extraction via word embeddings. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*.

Daniel Gordon, Aniruddha Kembhavi, Mohammad Rastegari, Joseph Redmon, Dieter Fox, and Ali Farhadi. 2018. IQA: Visual question answering in interactive environments. In *Computer Vision and Pattern Recognition*.

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*.

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 54th Annual Meeting of the Association for Computational Linguistics*.

Ayush Jain, Andrew Szot, and Joseph Lim. 2020. Generalization to new actions in reinforcement learning. In *Proceedings of Machine Learning Research*.

Bartosz Kostka, Jaroslaw Kwiecien, Jakub Kowalski, and Pawel Rychlikowski. 2017. Text-based adventures of the golovin ai agent. In *IEEE Conference on Computational Intelligence and Games*.

Angeliki Lazaridou, Anna Potapenko, and Olivier Tieleman. 2020. Multi-agent communication meets natural language: Synergies between functional and structural language learning. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

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 28th International Joint Conference on Artificial Intelligence*.

George A Miller. 1998. *WordNet: An electronic lexical database*. MIT press.

Karthik Narasimhan, Regina Barzilay, and Tommi Jaakkola. 2018. Grounding language for transfer in deep reinforcement learning. *Journal of Artificial Intelligence Research*, 63(1):849–874.

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

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *Open AI Blog*.

Karin Kipper Schuler. 2006. *VerbNet: A Broad-Coverage, Comprehensive Verb Lexicon*. Ph.D. thesis, University of Pennsylvania.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*.

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

Xingdi Yuan, Marc-Alexandre Côté, Alessandro Sordoni, Romain Laroche, Remi Tachet des Combes, Matthew Hausknecht, and Adam Trischler. 2019. Counting to explore and generalize in text-based games. *arXiv preprint arXiv:1806.11525*.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. SWAG: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Xuhui Zhou, Yue Zhang, Leyang Cui, and Dandan Huang. 2020. Evaluating commonsense in pre-trained language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

A Definitions of Action Templates

We manually defined 41 abstractive action templates in total, and Table 6 shows extended examples. We give definitions of all of the templates at https://github.com/kohilin/lgat.

B Masker Function Details

Input for LM masker. The input of the LM masker is an observation string suffixed with one of the verbs in the template. For example, in the case of GIVE with the observation "*You have a cup of water. A boy stands in front of you.*", the input for the n_1 slot is "*... front of you. You gave the*," and that for the n_2 slot is "*... front of you. You gave it to the*."

Mask for primitive actions. L-GAT allows an agent to produce primitive actions, related to movement and belongings, regardless of the outputs of the masker function. Specifically, we forcibly assign 1 for direction words (*up*, *down*, *north*, *south*, *east*, *west*, *northwest*, *northeast*, *southwest*, *southeast*) to the n_1 slot in the GO template and *all* to the n_1 slots in the GET and DROP templates.

C Experimental Details

Hyperparameters. Table 5 shows our experimental hyperparameters and values searched for in the hyperparameter search. We tuned the α decay and initial ϵ, minimum ϵ, and ϵ decay for each game and each method. The other parameters were determined by testing with detective, temple, and zork1, and we applied the same settings to all of the games.

Machine specifications. We ran our experiment on Red Hat Enterprise Linux with an Intel(R) Xeon(R) CPU E5-2690 v4 at 2.60 GHz with 500GB of RAM and a single GPU, an NVIDIA TESLA K80.

Training time. Training with one game took approximately 10 ˜15 hours for the L-GAT agents, and 4 ˜6 hours for the TDQN+ agents depending on the game.

Model size. The number of trainable parameters for the L-GAT agent was about 10 million, which slightly changed depending on the vocabulary size of the games.

Type	Name	Value	Search
State Encoder	Algorithm	GRU	-
	Embedding size	256	128, 256
	Layer	2	-
	Hidden units	256	128, 256
Tokenizer	Algorithm	BERT subwording	-
	Max length	256	50, 256
Q-function	Layer	2	-
	Hidden units	200	-
	Activation	ReLU	-
Environment	Training episodes	1000	-
	Max steps	100	-
	Batch size	10	-
Optimizer	Algorithm	Adam	-
	Initial α	1e-3	1e-2, 1e-3
	α decay	0.5 or None	-
	α decay interval	100 episodes	-
Experience Replay	Buffer	Prioritized	-
	Replay batch size	128	128, 256
	Replay interval	5 steps	-
	Memory capacity	10000	-
Exploration	Algorithm	ϵ-greedy	-
	Initial ϵ	1.0 or .5	-
	Minimum ϵ	.05, .1, .2, or .3	-
	ϵ decay	.9975, .995, or .99	-
	ϵ decay interval	1000 steps	-
Loss	Loss function	Smoothed L1	-
	Discount factor	.95	-
	Reward clipping	-10, 10	-

Table 5: Hyperparameter settings and values in hyperparameter search.

Walkthrough experiment. In the experiment, we investigated how many walkthrough actions were covered by L-GAT. The same transition can be possible with different action strings. Therefore, for each state, we tried all generable strings of L-GAT and checked that at least one of them could produce the same transition with the gold walkthrough action. To judge if two transitions were the same or not, we used the game state hash provided by Jericho. Several walkthrough actions did not change the world state hash. We excluded such no-effect walkthrough actions from the evaluation since completely different actions were classified as replaceable for them.

D Learning Curves of Each Game

Figure 6 shows learning curves of the L-GAT and TDQN+ agents in each game.

E Game play Example in zork1

Table 7 shows the best play trajectory of our L-GAT agent in zork1.

	f	$r_{(v)}$	$r_{(n_1)}$	$r_{(p)}$	$r_{(n_2)}$
L-GAT	ASK	PREDICATE	PATIENT	PREPOSITION	THEME
FrameNet	Questioning	-	Addressee	-	Topic
VerbNet	inquire-37.1.2	-	Patient	-	Theme
WordNet	-	ask.v.05	people.n.01, person.n.01	about	object.n.01, ...
L-GAT	ATTACH	PREDICATE	THEME	PREPOSITION	DESTINATION
FrameNet	Attaching	-	Items	-	Goal
VerbNet	shake-22.3-2-1	-	Theme	-	Destination
WordNet	-	attach.v.01	attachment.n.04	to	object.n.01
L-GAT	ATTACK	PREDICATE	PATIENT	PREPOSITION	INSTRUMENT
FrameNet	Attack	-	Assailant	-	Weapon
VerbNet	-	-	Patient	-	Instrument
WordNet	-	attack.v.01, kill.v.01	monster.n.01, ...	with	weapon.n.01
L-GAT	BREAK	PREDICATE	THEME	NULL	NULL
FrameNet	Cause_to_fragment	-	Whole_patient	-	-
VerbNet	break-45.1	-	Theme	-	-
WordNet	-	break.v.02	object.n.01	-	-
L-GAT	BUY	PREDICATE	THEME	NULL	NULL
FrameNet	Commerce_buy	-	Goods	-	-
VerbNet	get-13.5.1	-	Theme	-	-
WordNet	-	buy.v.01	object.n.01	-	-
L-GAT	CHECK	PREDICATE	LOCATION	NULL	NULL
FrameNet	Scrutiny	-	Ground	-	-
VerbNet	investigate-35.4	-	Location	-	-
WordNet	-	check.v.01, examine.v.02	entity.n.01	-	-
L-GAT	CLEAN	PREDICATE	THEME	NULL	NULL
FrameNet	Emptying	-	Theme, Source	-	-
VerbNet	clear-10.3	-	Theme	-	-
WordNet	-	clean.v.01	area.n.01, instrumentality.n.03	-	-
L-GAT	CLOSE	PREDICATE	THEME	NULL	NULL
FrameNet	Closure	-	Container_portal	-	-
VerbNet	-	-	Theme	-	-
WordNet	-	close.v.01	window.n.01, ...	-	-
L-GAT	COOK	PREDICATE	THEME	NULL	NULL
FrameNet	Cooking_creation	-	Produced_food	-	-
VerbNet	cooking-45.3	-	Theme	-	-
WordNet	-	make.v.03, cook.v.01	food.n.01	-	-
L-GAT	CUT	PREDICATE	PATIENT	NULL	NULL
FrameNet	Cutting	-	Item	-	-
VerbNet	cut-21.1-1	-	Patient	-	-
WordNet	-	cut.v.01	food.n.01, hair.n.01	-	-
L-GAT	DROP	PREDICATE	THEME	NULL	NULL
FrameNet	Abandonment	-	Theme	-	-
VerbNet	leave-51.2	-	Theme	-	-
WordNet	-	drop.v.01	physical entity.n.01	-	-
L-GAT	EAT	PREDICATE	PATIENT	NULL	NULL
FrameNet	Ingestion	-	Ingestibles	-	-
VerbNet	eat-39.1-1	-	Patient	-	-
WordNet	-	eat.v.01	food.n.01	-	-
L-GAT	FILL	PREDICATE	DESTINATION	PREPOSITION	THEME
FrameNet	Filling	-	Goal	-	Theme
VerbNet	fill-9.8-1	-	Destination	-	Theme
WordNet	-	fill.v.01	container.n.01	with	fluid.n.01
L-GAT	GET	PREDICATE	THEME	NULL	NULL
FrameNet	Taking	-	Theme	-	-
VerbNet	get-13.5.1	-	Theme	-	-
WordNet	-	get.v.01, take.v.01	physical entity.n.01	-	-

Table 6: Extended examples of abstractive action templates in L-GAT.

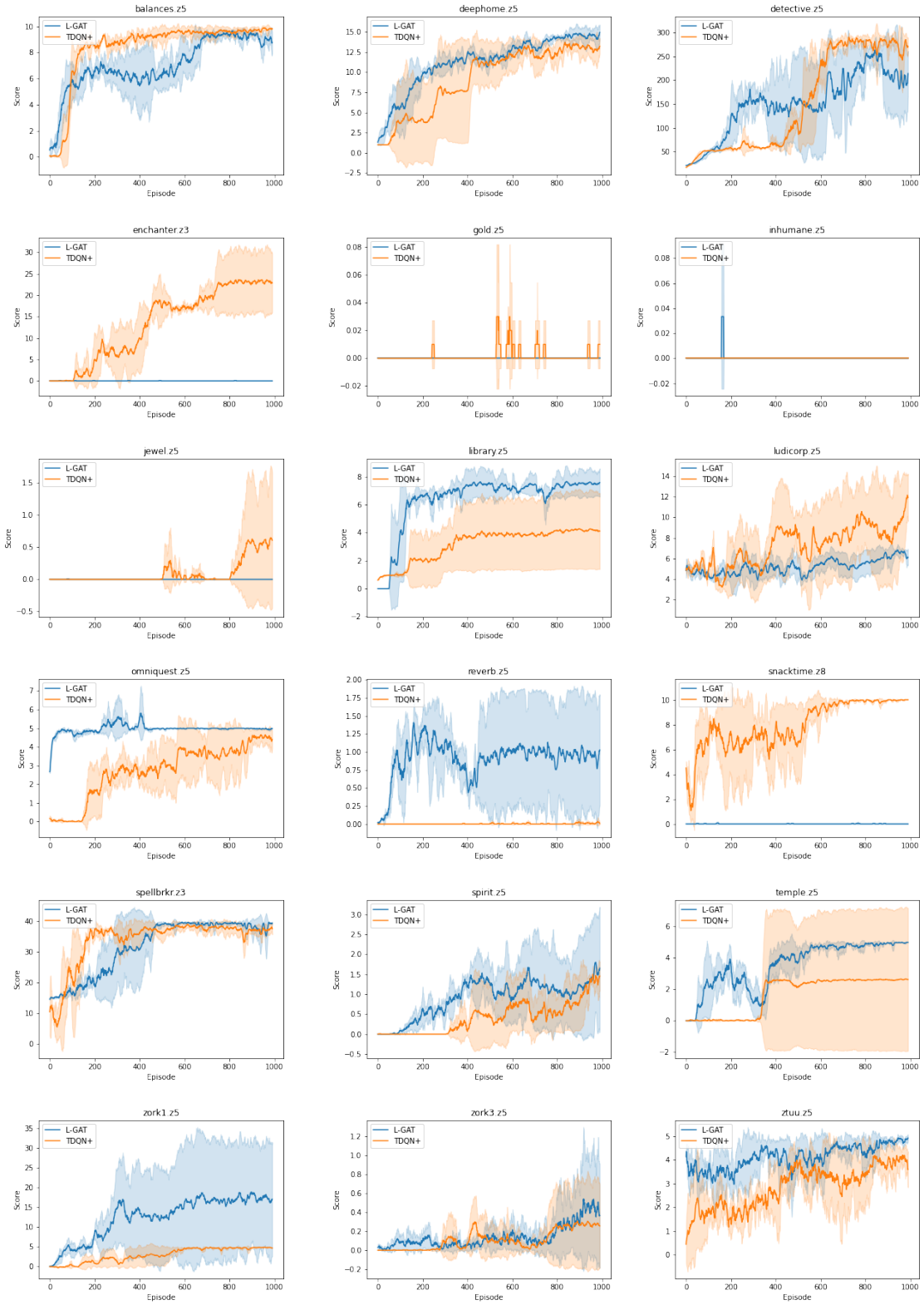

Figure 6: Learning curves of L-GAT (blue) and TDQN+ (orange).

Observation	Action	Score
West of House You are standing in an open field west of a white house, with a boarded front door. There is a small mailbox ...	examine mailbox	0
The small mailbox is closed.	go north	0
North of House You are facing the north side of a white house. There is no door here, and all the windows are boarded ...	go east	0
Behind House You are behind the white house. A path leads into the forest to the east. In one corner of the house there ...	go east	0
Clearing You are in a small clearing in a well marked forest path that extends to the east and west.	go west	0
Behind House	go east	0
Clearing	go east	0
Canyon View You are at the top of the Great Canyon on its west wall. From here there is a marvelous view of the ...	go west	0
Forest This is a dimly lit forest, with large trees all around.	go west	0
Forest This is a forest, with trees in all directions. To the east, there appears to be sunlight.	go east	0
Forest Path This is a path winding through a dimly lit forest. The path heads north-south here. One particularly large tree ...	go up	0
Up a Tree You ... Beside you on the branch is a small bird's nest. In the bird's nest is a large egg encrusted with precious ...	eat egg	5
(Taken) I don't think that the jewel-encrusted egg would agree with you.	drop all	5
jewel-encrusted egg: The egg falls to the ground and springs open, seriously damaged. There is a golden clockwork canary ...	get bird	5
The songbird is not here but is probably nearby. You hear in the distance the chirping of a song bird.	get all	5
bird's nest: Taken. You hear in the distance the chirping of a song bird.	drop all	5
bird's nest: The bird's nest falls to the ground.	go up	5
You cannot climb any higher.	go down	5
Forest Path There is a bird's nest here. There is a somewhat ruined egg here. There is a golden clockwork canary nestled in ...	go south	5
North of House	go east	5
Behind House	open window	5
With great effort, you open the window far enough to allow entry.	open white	15
Kitchen You are in the kitchen of the white house. A table seems to have been used recently for the preparation of food ...	go west	15
Living Room You are in the living room. There is a doorway to the east, a wooden door with strange gothic lettering to the ...	get all	15
trophy case: The trophy case is securely fastened to the wall. sword: Taken. brass lantern: Taken. carpet: The rug is ...	drop all	15
brass lantern: Dropped. sword: Dropped.	ignite lantern	15
What do you want to ignite the lantern with?	push lantern	15
Pushing the brass lantern doesn't seem to work.	light lantern	15
(Taken) The brass lantern is now on.	drop all	15
brass lantern: Dropped.	go east	15
Kitchen On the table is an elongated brown sack, smelling of hot peppers. A bottle is sitting on the table. The glass bottle ...	go west	15
Living Room There is a brass lantern (battery-powered) here. There is a sword here.	get all	15
brass lantern: Taken. sword: Taken. trophy case: The trophy case is securely fastened to the wall. carpet: The rug is ...	drop lantern	15
Dropped.	drop all	15
sword: Dropped.	go east	15
Kitchen On the table is an elongated brown sack, smelling of hot peppers. A bottle is sitting on the table. The glass bottle ...	open sack	15
Opening the brown sack reveals a lunch, and a clove of garlic.	take all	15
brown sack: Taken. glass bottle: Taken.	go west	15
Living Room There is a sword here. There is a brass lantern (battery-powered) here.	drop all	15
glass bottle: Dropped. brown sack: Dropped.	move rug	15
With a great effort, the rug is moved to one side of the room, revealing the dusty cover of a closed trap door.	open cover	15
The door reluctantly opens to reveal a rickety staircase descending into darkness.	ask case	15
You can't talk to the trophy case!	wait	15
Time passes...	get all	15
brown sack: Taken. glass bottle: Taken. sword: Taken. brass lantern: Taken. trophy case: The trophy case is securely ...	go down	15
The trap door crashes shut, and you hear someone barring it. Cellar You are in a dark and damp cellar with a narrow ...	-	40

Table 7: Best play trajectory of L-GAT agent in zork1 until reaching best score of 40. For simplicity, we excluded steps with no-effect actions in this table.