Text-based RL Agents with Commonsense Knowledge: New Challenges, Environments and Baselines

Anonymous EMNLP submission

Abstract

Text-based games have emerged as an important test-bed for Reinforcement Learning (RL) research, requiring RL agents to combine grounded language understanding with sequential decision making. In this paper, we examine the problem of infusing RL agents with commonsense knowledge. Commonsense would allow agents to efficiently act in the world by pruning out implausible actions, and to perform look-ahead planning to determine how current actions might affect future world states. We design new text-based gaming environments called `TextWorld Commonsense` (`TWC`) for training and evaluating RL agents with a specific kind of commonsense knowledge about objects, their attributes, and affordances. We also introduce several baseline RL agents which track the sequential context and dynamically retrieve the relevant commonsense knowledge from ConceptNet. We show that our agents act efficiently (fewer moves) and achieve better scores when we incorporate commonsense, and that the learned policies can be transferred to other instances in `TWC`.

1 Introduction

Over the years, simulation environments have been used extensively to drive advances in reinforcement learning (RL). A recent environment that has received much attention is TextWorld (`TW`) (Côté et al., 2018), where an agent must interact with an external environment to achieve goals while maximizing reward - all of this using only the modality of text. TextWorld and similar text-based environments seek to bring advances in grounded language understanding in a sequential decision making setup.

While existing text-based games are valuable for RL research, they fail to test a key aspect of human intelligence: commonsense. Humans capitalize on commonsense (background) knowledge about entities – properties, spatial relations, events, causes and effects, and other social conventions – while interacting with the world (Mccarthy, 1960; Winograd, 1972; Davis and Marcus, 2015).

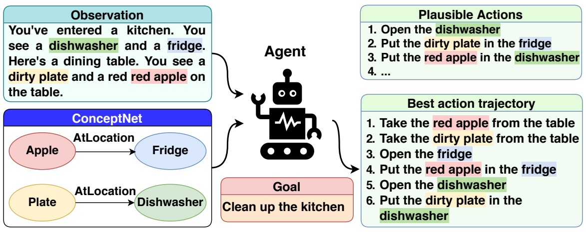

Figure 1: An illustration of a `TWC` game. The agent is given an initial observation (top left) and has to produce the list of actions (bottom right) that are necessary to achieve this goal (bottom center) using relevant commonsense knowledge from ConceptNet (bottom left).

Motivated by this, we propose novel text-based environments, `TextWorld Commonsense` (or `TWC`), where the agent is expected to use commonsense knowledge stored in knowledge bases such as *ConceptNet* (Liu and Singh, 2004; Speer et al., 2017) to act efficiently. `TWC` is a sandbox environment similar to `TextWorld`, where the agent has to clean up a house. Efficiently achieving goals in this environment requires commonsense knowledge about objects, their properties, locations, and affordances. Efficient use of commonsense would allow the agent to select correct and applicable actions at each step: i.e., improve sample efficiency by reducing exploration; as well as to perform look-ahead planning to determine how current actions might affect future world states (Juba, 2016). Figure 1 presents a running example from `TWC` that illustrates this: in the figure, the additional knowledge that must be utilized effectively by the agent is shown in the bottom left corner.

Building commonsense-based RL agents for text-based games is hard. The agent is required to accurately model the sequential context and track the state of the game. At the same time, the agent must also be able to dynamically retrieve the relevant commonsense knowledge with high precision, and use it appropriately. In this paper, we present an agent that combines the game state with relevant commonsense knowledge. The agent tracks the state of the game using a high-level recurrent architecture over observation representations. Then, it dynamically retrieves the relevant commonsense based on the sequential context using a number of simple graph linking and neighborhood exploration techniques. Finally, it combines the game state with the retrieved commonsense subgraph using a co-attention mechanism.

We showcase improvements in the performance of our commonsense RL agents on `TWC` as they complete the house cleanup tasks and achieve a higher score (discounted reward from the environment) in fewer steps in comparison to a purely text-based model. Moreover, the RL agent with commonsense knowledge also achieves the best generalization to other game instances in `TWC`.

Contributions: (1) We propose the use of commonsense knowledge from external knowledge bases to make text-based RL agents more efficient; (2) To support more research in this area, we generate a new environment (`TWC`) that requires commonsense knowledge; (3) We propose a model that tracks sequential context, dynamically retrieves relevant commonsense knowledge and combines the context information with the commonsense knowledge for decision making; and (4) We show empirically that agents thus constituted are more efficient than purely text-based agents.

2 TextWorld Commonsense (TWC)

Commonsense can be defined very broadly and in various ways. However, in this paper, we mainly focus on commonsense knowledge that pertains to objects, their attributes, and affordances[1]. Several existing text-based games designed with TextWorld (Adhikari et al., 2020; Côté et al., 2018) severely restrict the amount and variety of external commonsense knowledge that an agent needs to know and exploit. Thus, in this paper, we create and present a new domain – `TextWorld Commonsense (TWC)` – by reusing the TextWorld engine as described below in order to generate text-based environments where RL agents need to effectively retrieve and use commonsense knowledge.

[1]Gibson in his seminal work (Gibson, 1978) refers to affordance as “properties of an object [...] that determine what actions a human can perform on them”.

2.1 Constructing TWC

We built the `TWC` domain as a house clean-up environment where the agent is required to obtain and use knowledge about typical objects in the house, their properties, and expected location from a commonsense knowledge base. The house is initialized with random placement of objects in various locations. The agent's high level goal is to tidy up the house by putting objects in their `commonsense` locations. This high level goal may consist of multiple sub-goals. For example, for the sub-goal: *put the apple inside the refrigerator*, commonsense knowledge from ConceptNet such as (`Apple` → `AtLocation` → `Refrigerator`) can assist the agent.

Goal Sources: While our main objective was to create environments that require commonsense, we did not want to bias the environments towards any one of the existing commonsense knowledge bases. We additionally wanted to rule out the possibility of data leaks in situations where both the environment as well as the external knowledge came from the same part of a specific commonsense knowledge base (KB) like ConceptNet. For the construction of the `TWC` goal instances, we picked sources of information that were orthogonal to existing commonsense KBs. Specifically, we used: (1) the `picture dictionary from 7ESL`[2]; (2) the `British Council's vocabulary learning page`[3]; (3) the `English At Home vocabulary learning page`[4]; and (4) `ESOL courses`[5]. We collected vocabulary terms from these sources and manually aggregated this content in order to build a dataset that lists several kinds of objects that are typically found in a house environment. For each object, the dataset specifies a list of coherent plausible locations within the house.

Instance Construction: A `TWC` instance is sampled from this dataset, which includes a configuration of 8 room types and a total of more than 900 entities (Table 1). The environment includes three main kinds of entities: objects, supporters,

[2]https://7esl.com/picture-dictionary/

[3]https://learnenglish.britishcouncil.org/vocabulary/beginner-to-pre-intermediate

[4]https://www.english-at-home.com/vocabulary

[5]www.esolcourses.com/topics/household-home.html

	Count	Examples
Rooms	8	*kitchen, backyard*
Supporters/Containers	56	*dining table, wardrobe*
Unique Objects	190	*plate, dress*
Total Objects	872	*dirty plate, clean red dress*
Total Entities	928	*dirty plate, dining table*

Table 1: Statistics on the number of entities, supporters/containers, and rooms in the TWC domain.

	Correctness	Completeness
Rated Commonsense	669	47
Rated NOT Commonsense	31	253

Table 2: Statistics from the human annotations to verify TWC

and containers. Objects are entities that can be carried by the agent, whereas supporters and containers are furniture where those objects can be placed. Let o represent the object or entity in the house; r represent the room that the entity is typically found in; and l represent the location inside that room where the entity is typically placed. In our running example, o:apple is an entity, l:refrigerator is the container, and r:kitchen is the room. Via a manual verification process (which we elucidate next in Section 2.2) we ensure that the associations between entities, supporters/containers, and rooms reflect commonsense. As shown in Table 1, we collected a total of 190 objects from the aforementioned resources. We further expanded this list by manually annotating the objects with qualifying properties, which are usually adjectives from a predefined set (e.g., a shirt may have a color and a specific texture). This allows increasing the total pool of objects for generating TWC environments to more than 800.

2.2 Verifying TWC

In order to ensure that TWC indeed reflects commonsense knowledge, we set up two annotation tasks to verify the environment goals (i.e., goal triples of the form ⟨o, r, l⟩, where o stands for an object, r denotes a room, and l a location within that room, as defined in Section 2.1). The first task is meant to verify the correctness of the goals and evaluate whether the goal ⟨o, r, l⟩ triples make sense to humans. The second task is aimed at verifying completeness, i.e. that other triples in the environment do not make sense to humans.

Verifying Correctness: To test the correctness of our environments, we asked our human annotators to determine whether they would consider a given room-location combination in the goal ⟨o, r, l⟩ to be a reasonable place for the object o; if so, the instance was labeled positive, and negative otherwise. We collected annotations from $M = 10$ annotators, across a total of $N = 205$ unique ⟨o, r, l⟩ triples. Each annotator annotated 70 of these triples, and each triple was annotated by at least 3 distinct annotators. The annotators were not given any other biasing information, and all annotators worked independently. We found a heavy label bias in the annotations: more than 95% of all responses fall into the 'positive' nominal category leading to asymmetrically imbalanced marginals. In this case, standard inter-annotator agreement statistics like *Cohen's kappa, Fleiss' kappa* and *Krippendorff's alpha* are not reliable (Feinstein and Cicchetti, 1990). Thus, we simply show overall agreement of the annotators with TWC's goals in Table 2. The high agreement from the annotators demonstrates that the goal ⟨o, r, l⟩ triples reflect human commonsense knowledge.

Verifying Completeness: Similar to the above annotation exercise, we also asked human annotators to determine if a non-goal ⟨o, r, l⟩ triple made sense to them. In addition to the 70 triples mentioned above, each of the $M = 10$ annotators were asked to label as either positive or negative a set of 30 non-goal triples. In order to provide annotators with an informative set of non-goal ⟨o, r, l⟩ triples, we used GloVe (Pennington et al., 2014) to compute location embeddings for each location in TWC. For a given object o, a non-goal location l' was then selected among those most similar to the goal location l, according to the cosine similarity between the embeddings of l and l'. As before, each non-goal triple was assigned to at least 3 annotators from a set that comprises a total of 97 triples. As we see in Table 2, the annotators seldom find a hypothesized non-goal ⟨o, r, l⟩ triple as commonsensical.

Annotator Reliability: For our overall annotation exercise, we can report inter-annotator agreement statistics, as the overall annotation is no longer imbalanced in terms of label marginals. We report a *Krippendorff's alpha* (Krippendorff, 2018) $\alpha_K = 0.74$. This number is over the accepted range for agreement and shows that our annotators have a strong agreement when rating the triples.

2.3 TWC Games

We used the TextWorld engine to build a set of text-based games where the goal is to tidy up a house by putting objects in the goal locations specified in the TWC dataset. The games are grouped

	#objects	#Objects to find	#Rooms
Easy	1	1	1
Medium	2, 3	1, 2, 3	1
Hard	6, 7	5, 6, 7	1, 2

Table 3: Specification of `TWC` games

into three difficulty levels (easy, medium, and hard) depending on the total number of objects in the game, the number of objects that the agent needs to find (the remaining ones are already carried by the agent at the beginning of the game) and the number of rooms to explore. The values of these properties are randomly sampled from the ones listed in Table 3. For each difficulty level, we provide a training set and two test sets. The training sets were built out of $2/3$ of the unique objects reported in Table 1. For the first test set, we used the same set of objects as the training games to generate evaluation games. We call this set the *in* distribution test set. For the second test set, we employed the remaining $1/3$ objects to create test games. We call this set *out* of distribution test set. This allows us to investigate not only the capability of the agents to generalize within the same distribution of the training data, but also their ability to achieve generalization to unseen entities.

3 `TWC` Agents

Text-based games can be seen as partially observable Markov decision processes (POMDP) (Kaelbling et al., 1998) where the system dynamics are determined by an MDP, but the agent cannot directly observe the underlying state. The agent receives a reward at every time step and the agent's goal is to maximize the expected discounted sum of rewards. The `TWC` games allow the agent to perceive and interact with the environment via text. Thus, the observation at time step t, o_t is presented by the environment as a sequence of tokens ($o_t = \{o_t^1, \dots o_t^N\}$). Similarly, each action a is also denoted as a sequence of tokens $\{a^1, \dots, a^M\}$. The goal of this project is to test RL agents with commonsense. In this case, the agents also have access to a commonsense knowledge base, and are allowed to use it while selecting actions.

In order to model `TWC`, we design a framework that can (a) learn representations of various actions, (b) learn from sequential context, (c) dynamically retrieve the relevant commonsense knowledge, (d) integrate the retrieved relevant commonsense knowledge with the context, and (e) predict actions. A block diagram of the framework is shown in Figure 2. We describe the

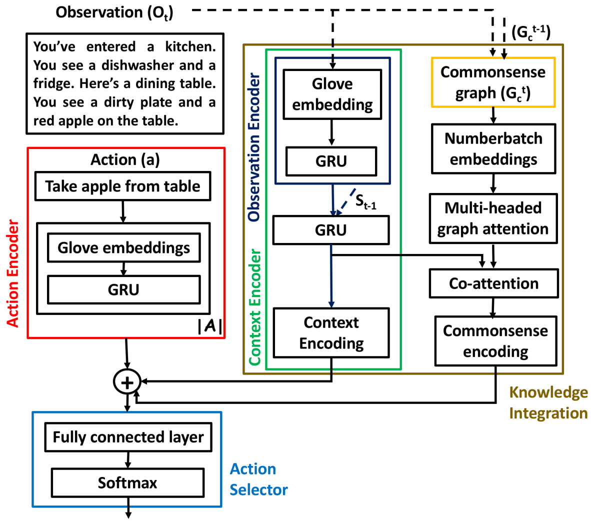

Figure 2: Overview of our framework's decision making at any given time step. The framework comprises of the following components (visually shown in color): (a) action encoder which encodes all admissible actions $a \in \mathscr{A}$, (b) observation encoder which encodes the observation o_t, (c) context encoder, which encodes the dynamic context C_t, (d) a dynamic common sense subgraph of ConceptNet G_C^t extracted by the agent, (e) a knowledge integration component, which combines the information from textual observations and the extracted common sense subgraph, and (f) an action selection module. $\oplus$ denotes the concatenation operator.

various components of our framework below.

3.1 Action and Observation Encoder

We learn representations of observations and actions by feeding them to a recurrent network. Given current observation o_t, we use pre-trained GloVe embeddings (Pennington et al., 2014) to represent o_t as a sequence of d-dimensional vectors $\mathbf{x}_t^1, \dots, \mathbf{x}_t^N$, where each $\mathbf{x}_t^k \in \mathbb{R}^d$ is the glove embedding of the k-th observed token o_t^k, $k = 1, \dots, N$. Then, a (bidirectional) GRU-based encoder (Cho et al., 2014) is used to process the sequence $\mathbf{x}_t^1, \dots, \mathbf{x}_t^N$. This gives us the representation of the current observation: $\mathbf{o}_t = \mathbf{h}_t^N$, where $\mathbf{h}_t^k = GRU(\mathbf{h}_t^{k-1}, \mathbf{x}_t^k)$, for $k = 1, \dots, N$. In a similar way, given the set A_t of admissible actions at time step t, we learn representations of each action $a \in A_t$.

3.2 Context Encoder

A key challenge for our RL agent is in modeling the context, i.e. the entire history of observations. We model the context using another recurrent encoder over the observation representations $\mathbf{o}_t$. We use a GRU network to encode the sequence of previous observations up to o_t into a vector $\mathbf{s}_t = GRU(\mathbf{s}_{t-1}, \mathbf{o}_t)$. We refer to s_t as the state vector, or the context encoding. The context encoding will be used in addition to the commonsense knowledge in the final action prediction.

Recent work (Talmor et al., 2018; Huang et al., 2019; Fadnis et al., 2019) has shown that while external knowledge can be useful, it must be balanced by the context-specific relevance of that new information. If this is not done properly, there is a high risk of overwhelming the agent with too much information, leading to poor decisions and performance. We, therefore, discuss several mechanisms to retrieve the relevant commonsense knowledge from an external knowledge source.

The commonsense knowledge retrieved by our agent is in the form of a graph. This is updated dynamically at each time step t with the resulting graph G_C^t. The graph is constructed by first mapping the textual observation o_t at time t to the external knowledge source – in our case, ConceptNet. This mapping is done by extracting and linking concepts mentioned in the observation text to ConceptNet. We used Spacy (Explosion, 2017) to extract noun chunks, and then performed a max sub-string match with all the concepts in ConceptNet. This results in a set of entities e_t for the observation o_t at time t.

Our next step is to construct G_C^t from the concepts extracted from the present observation e_t and the commonsense subgraph from the previous observations, G_C^{t-1}. We first combine the concepts from G_C^{t-1} and e_t to get E_t. E_t consists of all the concepts observed by the agent until time step t, including the description of the room, current observation from the environment, and the objects in the inventory. Given E_t, we describe three different techniques to automatically extract the commonsense graph G_t from external knowledge.

(1) Direct Connections: This is the baseline approach to construct G_C^t. We fetch direct links between each of the concepts in E_t from ConceptNet.

(2) Contextual Direct Connections: Since the goal of the agent is to clean up the house by putting `objects` into its appropriate `containers`, we hypothesize that adding links only between objects and containers may benefit the agent instead of links between all concepts as done by *Direct Connections*, as we might overwhelm the agent with noise. For example, assuming that the agent has seen the following: `clothes`, `apple`, `refrigerator`, and `washing machine`, the agent can benefit from edges between `objects` and `containers` such as: (1) `clothes` $\Rightarrow$ `washing machine`, and (2) `apple` $\Rightarrow$ `refrigerator`, rather than links between objects and between containers such as: (1) `washing machine` $\Rightarrow$ `refrigerator`, and (2) `apple` $\Rightarrow$ `clothes`. To accomplish this goal, we split the entities E_t into objects and containers. Since we know the inventory, the objects from the inventory in E_t constitutes objects and we consider the remaining as containers. We retain only the edges between objects and containers from ConceptNet.

(3) Neighborhood: The previous techniques focus only on connecting the links between observed concepts, E_t, from external knowledge. In addition to the direct relations, it may be beneficial to include concepts from external knowledge that is related to E_t but has not been directly observed from the game. Therefore, for each concept in E_t, we include all its neighboring concepts and associated links from the external knowledge.

3.3 Knowledge Integration

We enhance our text-based RL agent by allowing it to jointly contextualize information from both the commonsense subgraph as well as the observation representation. We call this step knowledge integration. In this step, we encode the retrieved commonsense graph using a graph encoder followed by a co-attention layer.

Graph encoder: The graph G_C^t is encoded as follows: First, we use pretrained KG embeddings (Numberbatch) to map the set of nodes $\mathcal{V}_t$ to a feature matrix $[\mathbf{e}_t^1, \ldots, \mathbf{e}_t^{|\mathcal{V}_t|}] \in \mathbb{R}^{f \times |\mathcal{V}_*^t|}$. Here, $\mathbf{e}_t^i \in \mathbb{R}^f$ is the (averaged) embedding of words in node $i \in \mathcal{V}_*^t$. Following (Lu et al., 2017), we also add a *sentinel* vector to allow the attention modules to not attend to any specific nodes in the subgraph. These node embeddings are updated at each time step by message passing between the nodes of G_c^t using Graph Attention Networks (GATs) (Veličković et al., 2018) to get $\{\mathbf{z}_t^1, \mathbf{z}_t^2 \cdots \mathbf{z}_t^{|\mathcal{V}_t|}\}$, using multi-head graph attention resulting in a final graph representation that better captures the conceptual relations between the nodes in the subgraph.

Co-Attention: In order to combine the observational context and the retrieved commonsense graph, we consider a bidirectional attention flow layer between these representations to recontextualize the graph for the current state of the game (Seo et al., 2016; Yu et al., 2018).

Similar to (Yu et al., 2018), we compute a similarity matrix $S \in \mathbb{R}^{N\times|\mathscr{V}_C^t|}$ between the context and entities in the extracted common sense subgraph using a trilinear function. In particular, the similarity between j^{th} token's context encoding $\mathbf{h}_t^j$ and i^{th} node encoding $\mathbf{z}_t^i$ in the commonsense subgraph is computed as: $S_{ij} = \mathbf{W}_0^T[\mathbf{z}_t^i;\mathbf{h}_t^j;\mathbf{z}_t^i \circ \mathbf{h}_t^j]$ where $\circ$ denotes element-wise product, ; denotes concatenation and $\mathbf{W}_0$ is a learnable parameter. We use the softmax function to normalize the rows of S and get the similarity function for the common-sense knowledge graph $\bar{S}_G$. Similarly, we use the softmax function over the column vectors to get a similar function for the context representation $\bar{S}_O$. The commonsense-to-context attention is calculated as $A = \bar{S}_G^T \cdot O$ and the context-to-common sense attention is calculated as $B = \bar{S}_G\bar{S}_O^T \cdot G$, where $G = [\mathbf{z}_t^1,\mathbf{z}_t^2,\cdots\mathbf{z}_t^{|\mathscr{V}_C^t|}]$ and $O = [\mathbf{h}_t^1,\mathbf{h}_t^2\cdots\mathbf{h}_t^N]$ are the commonsense graph and observation encodings. Finally, the attention vectors are combined together and the final graph encoding vectors $\mathbf{G}$ are calculated as $\mathbf{W}^\top[\mathbf{G};\mathbf{A};\mathbf{G}\circ\mathbf{A};\mathbf{G}\circ\mathbf{B}]$ where W is the learnable parameter.

Finally, we get the commonsense graph encoding $\mathbf{g}_i^t$ for each action $a_i \in A_t$ by applying a general attention over the nodes using the state vector and the action encoding $[\mathbf{s}_t;\mathbf{a}_i^t]$ (Luong et al., 2015). The attention score for each node is computed as $\alpha_i = [\mathbf{s}_t;\mathbf{a}_i^t]\mathbf{W}_g\mathbf{G}$, and the commonsense graph encoding for action $\mathbf{a}_i^t$ is given as $\mathbf{g}_i^t = \alpha_i^\top G$.

3.4 Action Selection

The action score for each action $\hat{a}_i^t$ is computed based on the context encoding $\mathbf{s}_t$, the commonsense graph encoding $\mathbf{g}_i^t$ and the action encoding $\mathbf{a}_i^t$. We concatenate these encoding vectors into a single vector $\mathbf{r}_i^t = [\mathbf{s}_t;\mathbf{g}_i^t;\mathbf{a}_i^t]$. Then, we compute probability score for each action $a_i \in A_t$ as

$$\mathbf{p}_t = softmax(W_1 \cdot ReLU(W_2 \cdot \mathbf{r}_t + \mathbf{b}_2) + \mathbf{b}_1)$$

where $W_1, W_2, \mathbf{b}_1$, and $\mathbf{b}_2$ are learnable parameters of the model. The final action chosen by the agent is then given by the one with the maximum probability score, namely $\hat{a}_t = \arg\max_i p_{t,i}$.

4 Experiments

In this section, we report the results of our experiments on the TWC games. We measure the performance of the various agents using the normalized score (score achieved ÷ maximum achievable score) and the number of steps taken. Each agent is trained for 100 episodes and the results are averaged over 3 runs. Following the winning strategy in the FirstTextWorld competition (Adolphs and Hofmann, 2019), we use the Advantage Actor-Critic framework (Mnih et al., 2016) to train the agents using reward signals from the training games. In our experiments, we use ConceptNet as the commonsense knowledge base.

4.1 Sample Efficient RL

We evaluate the framework shown in Figure 2 on the TWC cleanup games (as described in Section 2.3). For comparison, we consider a random agent that picks an action at each time step randomly. We consider two types of RL agents based on the amount of information available to them. The `Text-based` RL agent has access to the textual description of the current state of the game provided by the TextWorld environment, whereas, `Commonsense-based` RL has access to both the textual information and ConceptNet. Our goal in these experiments is to show that the commonsense-based RL agent has noticeable advantages over the text-based RL agent. We are interested in a sample efficient exploration where the external knowledge from the commonsense subgraph is used to prune out the reward-poor (state, action) pairs and focus on the reward-rich pairs.

To show the improvement on this front, we focus on the average number of steps taken by the agents to achieve the reported score. Figure 3 shows the performance evaluation of the RL agents with `Text` and `Text+Commonsense` on the three difficulty levels in TWC games. We see that the commonsense-based RL agent performs better than the random and text-based RL agents in the easy and medium level games. This is not surprising, as these instances involve picking an object and placing it in a container in the same room. Both the text-based and commonsense RL agents struggle in the hard level, as these games have more than one room to explore. On the other hand, we notice that the average steps taken by the commonsense-based RL agent are noticeably lower than the other agents: it efficiently uses commonsense knowledge to rule out implausible actions. Further exploration of efficient ways to combine commonsense, observations, and feedback from the environment will prove beneficial for efficient sample exploration of this problem.

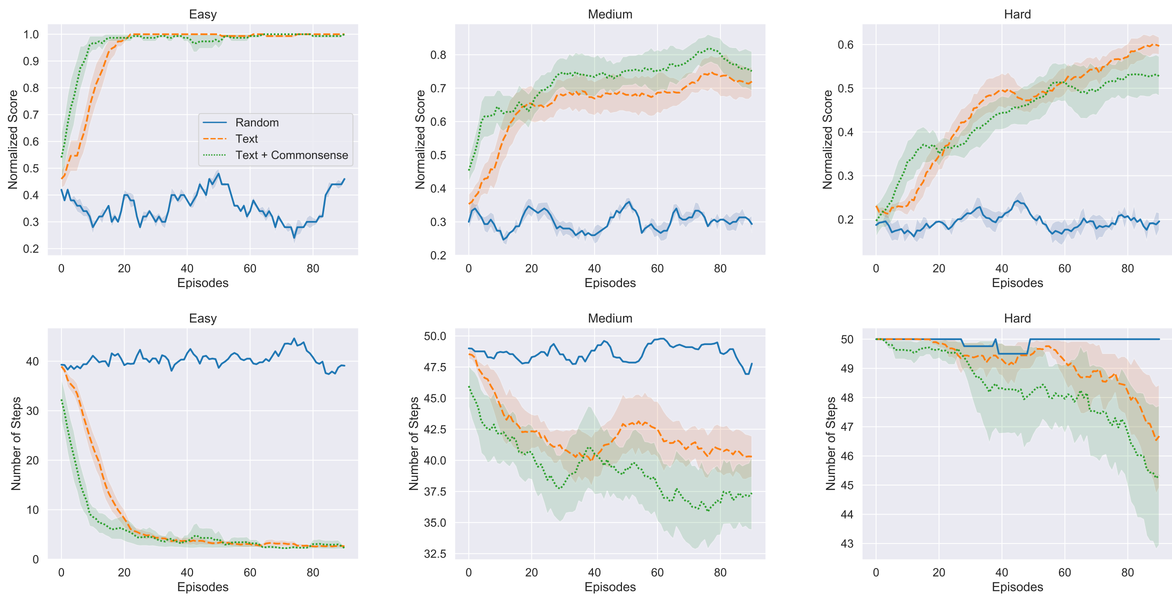

Figure 3: Performance evaluation (showing mean and standard deviation averaged over 3 runs) for the three difficulty levels: Easy (left), Medium (middle), Hard (right) using normalized score and the number of steps taken.

4.2 Generalization

We evaluated the generalization error of the agents on the test sets generated along with the training `TWC` games. Table 4 reports the results both for the test games that belong to the same distribution used at training time, and for test games that were generated from a different set of entities. For each difficulty level, we report: the normalized score achieved by the agent; the number of steps that the agent needed to reach the goal; and the optimal number of steps to solve the game. The optimal number of steps were computed by considering the objects already in the agent's possession (and not), the number of objects to "place" (goals), and the number of rooms in the instance. We do not currently consider distractor objects – i.e., objects that are not part of a goal. The commonsense-enhanced agent outperforms the text-only agent in all cases. There is also a clear distinction between the *in* distribution and *out* of distribution instances for the easy and medium levels. Interestingly, for the hard level, the agents struggle with both settings – we surmise that this is a result of the additional complexity of having to navigate between rooms. Finally, we also point out the vast gulf between the agents' current performance, versus the optimal number of steps: this attests to the promise of `TWC` as a domain for an active research study.

4.3 Commonsense Retrieval

In this section, we describe the behavior of our commonsense-based RL agent based on commonsense graphs generated by three different strategies: (1) Direct Connections (`DC`), (2) Contextual Direct Connections (`CDC`), and (3) Neighborhood (Section 3.2). The comparison of the agent's performance is shown in Figure 4. The results show that `CDC` performs the best, particularly in comparison to `DC`. Unlike `DC` that includes all the links between observed concepts from ConceptNet, `CDC` restricts links to those between observed objects and *containers*. This selection of relevant links from ConceptNet significantly improves the performance of the agent.

The commonsense graph generated for `DC` and `CDC` is comprised of only the observed concepts. The `Neighborhood` technique, however, also includes unobserved concepts that are one-hop away from the observed concepts. Unfortunately, due to the enormity of ConceptNet, each concept can introduce approximately 40 neighboring concepts on average. This introduces more noise for the agent, and hence as shown in Figure 4 the performance drops. This follows the same trend as work that uses neighborhood graphs for other NLP tasks (Wang et al., 2019). However, we believe that careful inclusion of relevant unobserved concepts and links can improve performance: this is our future work. We present more results and analysis in the supplementary file.

5 Related Work

RL Environments and TextWorld: Games are a rich domain for studying grounded language

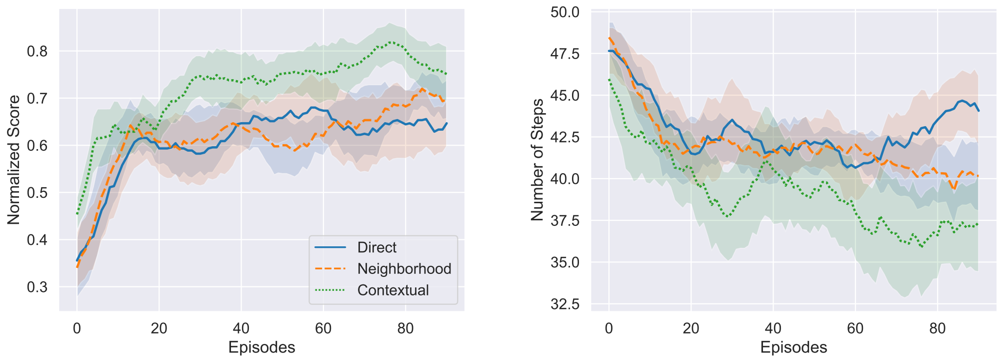

Figure 4: Performance evaluation for the medium level games (showing mean and standard deviation averaged over 3 runs) with the different techniques for the commonsense sub-graph extraction.

		Easy			Medium			Hard		
		Opt. #Steps	**#Steps**	**Norm. Score**	**Opt. #Steps**	**#Steps**	**Norm. Score**	**Opt. #Steps**	**#Steps**	**Norm. Score**
IN	**Text**	2.000 ± 0.000	15.787 ± 8.019	0.920 ± 0.040	3.600 ± 0.548	70.640 ± 7.990	0.747 ± 0.093	15.000 ± 2.000	100.000 ± 0.000	0.393 ± 0.049
	+Commonsense		$\mathbf{3.760 \pm 0.781}$	$\mathbf{1.000 \pm 0.000}$		$\mathbf{67.267 \pm 5.029}$	$\mathbf{0.780 \pm 0.026}$		$\mathbf{95.627 \pm 3.898}$	$\mathbf{0.583 \pm 0.072}$
OUT	**Text**	2.000 ± 0.000	26.667 ± 5.158	0.887 ± 0.076	4.400 ± 1.140	95.067 ± 1.686	0.530 ± 0.020	14.600 ± 2.673	100.000 ± 0.000	0.220 ± 0.053
	+Commonsense		$\mathbf{9.587 \pm 3.654}$	$\mathbf{0.987 \pm 0.023}$		$\mathbf{83.673 \pm 5.581}$	$\mathbf{0.650 \pm 0.098}$		$\mathbf{99.307 \pm 1.201}$	$\mathbf{0.360 \pm 0.079}$

Table 4: Generalization results for within distribution (IN) and out-of-distribution (OUT) games

and how information from text can be utilized in control. Recent work has explored text-based RL games to learn strategies for Civilization II (Branavan et al., 2012), multi-user dungeon games (Narasimhan et al., 2015), etc. Our work builds on `TextWorld` (Côté et al., 2018). A recent line of work on TextWorld learns symbolic (typically graphical) representations of the agent's belief. Notably, Ammanabrolu and Riedl (2019) proposed *KG-DQN* and Adhikari et al. (2020) proposed *GATA*; both represent the game state as a belief graph. This graph is used to prune the action space, enabling efficient exploration.

External Knowledge for Efficient RL: Garnelo et al. (2016) propose *Deep Symbolic RL*, which combines aspects of symbolic AI with neural networks and RL as a way to introduce commonsense priors. There has also been work on *policy transfer* (Bianchi et al., 2015), which studies how knowledge acquired in one environment can be re-used in another environment; and *experience replay* (Wang et al., 2016; Lin, 1992, 1993) which studies how an agent's previous experiences can be stored and then later reused. In this paper, we use commonsense knowledge as a way to improve sample efficiency in text-based RL agents. To the best of our knowledge, there is no prior work that *practically* explores how commonsense can be used to make RL agents more efficient. The most relevant prior work is by Martin et al. (2018), who use commonsense rules to build agents that can play tabletop role-playing games. However, unlike our work, the commonsense rules in this work are manually engineered and fixed.

Leveraging Commonsense: Recently, there has been a lot of work in NLP to utilize commonsense for QA, NLI, etc. (Sap et al., 2019; Talmor et al., 2018). Many of these approaches seek to effectively utilize ConceptNet by reducing the noise retrieved from it (Lin et al., 2019; Kapanipathi et al., 2020). This is also a key challenge in `TWC`.

6 Conclusion

We proposed the novel problem of using commonsense knowledge to build efficient RL agents for text-based games and created new environments (`TWC`) to test these agents in a home setting. We also introduced a new technique which tracks the state of the world, uses the sequential context to dynamically retrieve the relevant commonsense knowledge from a knowledge graph, and then combines the state information with the retrieved commonsense knowledge to act in the world. Our commonsense agents achieve their goals with greater efficiency and less exploration as compared to a text only model, thus showing the value of our new environments and models. We invite the research community to test their commonsense RL agents on our environments.

Replicability: As part of our contributions, we will release `TWC`; the game instances used for training and evaluating our models; the human annotations; and the code to generate the arbitrarily complex text-based games requiring commonsense knowledge.

References

Ashutosh Adhikari, Xingdi Yuan, Marc-Alexandre Côté, Mikuláš Zelinka, Marc-Antoine Rondeau, Romain Laroche, Pascal Poupart, Jian Tang, Adam Trischler, and William L Hamilton. 2020. Learning dynamic knowledge graphs to generalize on text-based games. *arXiv preprint arXiv:2002.09127*.

Leonard Adolphs and Thomas Hofmann. 2019. Ledeepchef: Deep reinforcement learning agent for families of text-based games. *ArXiv*, abs/1909.01646.

Prithviraj Ammanabrolu and Mark Riedl. 2019. Playing text-adventure games with graph-based deep reinforcement learning. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3557–3565.

Reinaldo AC Bianchi, Luiz A Celiberto Jr, Paulo E Santos, Jackson P Matsuura, and Ramon Lopez de Mantaras. 2015. Transferring knowledge as heuristics in reinforcement learning: A case-based approach. *Artificial Intelligence*, 226:102–121.

SRK Branavan, David Silver, and Regina Barzilay. 2012. Learning to win by reading manuals in a monte-carlo framework. *Journal of Artificial Intelligence Research*, 43:661–704.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734. ACL.

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

Ernest Davis and Gary Marcus. 2015. Commonsense reasoning and commonsense knowledge in artificial intelligence. *Communications of the ACM*, 58(9):92–103.

AI Explosion. 2017. spacy-industrial-strength natural language processing in python. *URL: https://spacy.io*.

Kshitij Fadnis, Kartik Talamadupula, Pavan Kapanipathi, Haque Ishfaq, Salim Roukos, and Achille Fokoue. 2019. Heuristics for Interpretable Knowledge Graph Contextualization. *arXiv preprint arXiv:1911.02085*.

Alvan R Feinstein and Domenic V Cicchetti. 1990. High agreement but low kappa: I. the problems of two paradoxes. *Journal of clinical epidemiology*, 43(6):543–549.

Marta Garnelo, Kai Arulkumaran, and Murray Shanahan. 2016. Towards deep symbolic reinforcement learning. *arXiv preprint arXiv:1609.05518*.

James J Gibson. 1978. The ecological approach to the visual perception of pictures. *Leonardo*, 11(3):227–235.

Lifu Huang, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Cosmos qa: Machine reading comprehension with contextual commonsense reasoning. *arXiv preprint arXiv:1909.00277*.

Brendan Juba. 2016. Integrated common sense learning and planning in pomdps. *The Journal of Machine Learning Research*, 17(1):3276–3312.

Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. 1998. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134.

Pavan Kapanipathi, Veronika Thost, Siva Sankalp Patel, Spencer Whitehead, Ibrahim Abdelaziz, Avinash Balakrishnan, Maria Chang, Kshitij Fadnis, Chulaka Gunasekara, Bassem Makni, Nicholas Mattei, Kartik Talamadupula, and Achille Fokoue. 2020. Infusing knowledge into the textual entailment task using graph convolutional networks. *AAAI*.

Klaus Krippendorff. 2018. *Content analysis: An introduction to its methodology*. Sage publications.

Bill Yuchen Lin, Xinyue Chen, Jamin Chen, and Xiang Ren. 2019. Kagnet: Knowledge-aware graph networks for commonsense reasoning. *arXiv preprint arXiv:1909.02151*.

Long-Ji Lin. 1992. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3-4):293–321.

Long-Ji Lin. 1993. Reinforcement learning for robots using neural networks. Technical report, Carnegie-Mellon Univ Pittsburgh PA School of Computer Science.

Hugo Liu and Push Singh. 2004. Conceptnet—a practical commonsense reasoning tool-kit. *BT technology journal*, 22(4):211–226.

Jiasen Lu, Caiming Xiong, Devi Parikh, and Richard Socher. 2017. Knowing when to look: Adaptive attention via a visual sentinel for image captioning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 375–383.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the*

2015 Conference on Empirical Methods in Natural Language Processing, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Lara J. Martin, Srijan Sood, and Mark Riedl. 2018. Dungeons and dqns: Toward reinforcement learning agents that play tabletop roleplaying games. In *Proceedings of the Joint Workshop on Intelligent Narrative Technologies and Workshop on Intelligent Cinematography and Editing co-located with 14th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, INTWICED@AIIDE 2018, Edmonton, Alberta, Canada, November 13-14, 2018*, volume 2321 of *CEUR Workshop Proceedings*. CEUR-WS.org.

John W. Mccarthy. 1960. Programs with common sense. In *Teddington Conference on the Mechanization of Thought Processes*.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937.

Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. 2015. Language understanding for text-based games using deep reinforcement learning. *arXiv preprint arXiv:1506.08941*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543. ACL.

Maarten Sap, Ronan Le Bras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A Smith, and Yejin Choi. 2019. Atomic: An atlas of machine commonsense for if-then reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3027–3035.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Robert Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *AAAI*, pages 4444–4451.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2018. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro LiÃš, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations*.

Xiaoyan Wang, Pavan Kapanipathi, Ryan Musa, Mo Yu, Kartik Talamadupula, Ibrahim Abdelaziz, Maria Chang, Achille Fokoue, Bassem Makni, Nicholas Mattei, et al. 2019. Improving natural language inference using external knowledge in the science questions domain. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7208–7215.

Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. 2016. Sample efficient actor-critic with experience replay. *arXiv preprint arXiv:1611.01224*.

Terry Winograd. 1972. Understanding natural language. *Cognitive psychology*, 3(1):1–191.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*.