LANGUAGE LEARNING IN INTERACTIVE ENVIRONMENTS

A Dissertation
Presented to
The Academic Faculty

By

Prithviraj Venkata Ammanabrolu

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Interactive Computing

Georgia Institute of Technology

August 2021

LANGUAGE LEARNING IN INTERACTIVE ENVIRONMENTS

Committee Members:

Dr. Mark O. Riedl, Advisor
School of Interactive Computing
Georgia Institute of Technology

Dr. Charles Isbell
College of Computing
Georgia Institute of Technology

Dr. Devi Parikh
School of Interactive Computing
Georgia Institute of Technology

Dr. Matthew Hausknecht
Micosoft Research

Dr. Jason Weston
Facebook AI Research
and New York University

Date: July 14, 2021

It is pitch black. You are likely to be eaten by a grue.

Zork

To Amma and Nanna.

ACKNOWLEDGEMENTS

[REDACTED]

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Natural language communication has long been considered a defining characteristic of human intelligence. I am motivated by the question of how learning agents can understand and generate contextually relevant natural language in service of achieving a goal. In pursuit of this objective, I have been studying Interactive Narratives, or text-adventures: simulations in which an agent interacts with the world purely through natural language—"seeing" and "acting upon" the world using textual descriptions and commands. These games are usually structured as puzzles or quests in which a player must complete a sequence of actions to succeed. My work studies two closely related aspects of Interactive Narratives: game-playing and game generation—each presenting its own set of unique challenges. **Structured contextualization, in the form of knowledge graphs, improves situated natural language generation and understanding in interactive environments as evaluated by (1) the ability to operate in textual worlds, and (2) the perceived coherence and creativity of procedurally generated language-based environments.**

Game-playing presents three challenges: (1) Knowledge representation—an agent must maintain a persistent memory of what it has learned through its experiences with a partially observable world; (2) Commonsense reasoning to endow the agent with priors on how to interact with the world around it; and (3) Scaling to effectively explore combinatorially-sized natural language state-action spaces. On the other hand, game generation can be split into two complementary considerations: (1) World generation, or the problem of creating a world that defines the limits of the actions an agent can perform; and (2) Quest generation, i.e. defining actionable objectives grounded in a given world. I will present my work thus far—showcasing how structured, interpretable data representations in the form of knowledge graphs aid in each of these tasks—in addition to proposing how exactly these two aspects of Interactive Narratives can be combined to improve language learning across this board of challenges.

CHAPTER 1
WHY INTERACTIVE ENVIRONMENTS?

Natural language communication has long been considered a defining characteristic of human intelligence. In humans, this communication is grounded in experience and real world context—"what" we say or do depends on the current context around us and "why" we say or do something draws on commonsense knowledge gained through experience. So how do we imbue learning agents with the ability to understand and generate contextually relevant natural language in service of achieving a goal?

Two key components in creating such agents are interactivity and environment grounding, shown to be vital parts of language learning in humans. Humans learn various skills such as language, vision, motor skills, etc. more effectively through interactive media (Feldman and Narayanan 2004; Barsalou 2008). In the realm of machines, interactive environments have served as cornerstones in the quest to develop more robust algorithms for learning agents across many machine learning sub-communities. Environments such as the Atari Learning Environment (Bellemare *et al.* 2013) and Minecraft Malmo (Johnson *et al.* 2016) have enabled the development of game agents that perform complex tasks while operating on raw video inputs, and more recently THOR (Kolve *et al.* 2017) and Habitat (Manolis Savva* *et al.* 2019) attempt to do the same with embodied agents in simulated 3D worlds.

Despite such progress in modern machine learning and natural language processing, agents that can communicate with humans (and other agents) through natural language in pursuit of their goals are still primitive. One possible reason for this is that many datasets and tasks used for Natural Language Processing (NLP) are static, not supporting interaction and language grounding (Feldman and Narayanan 2004; Barsalou 2008; Brooks 1991; Mikolov *et al.* 2016; Gauthier and Mordatch 2016; Lake *et al.* 2017). In other words, there has been a void for such interactive environments for purely language-oriented tasks.

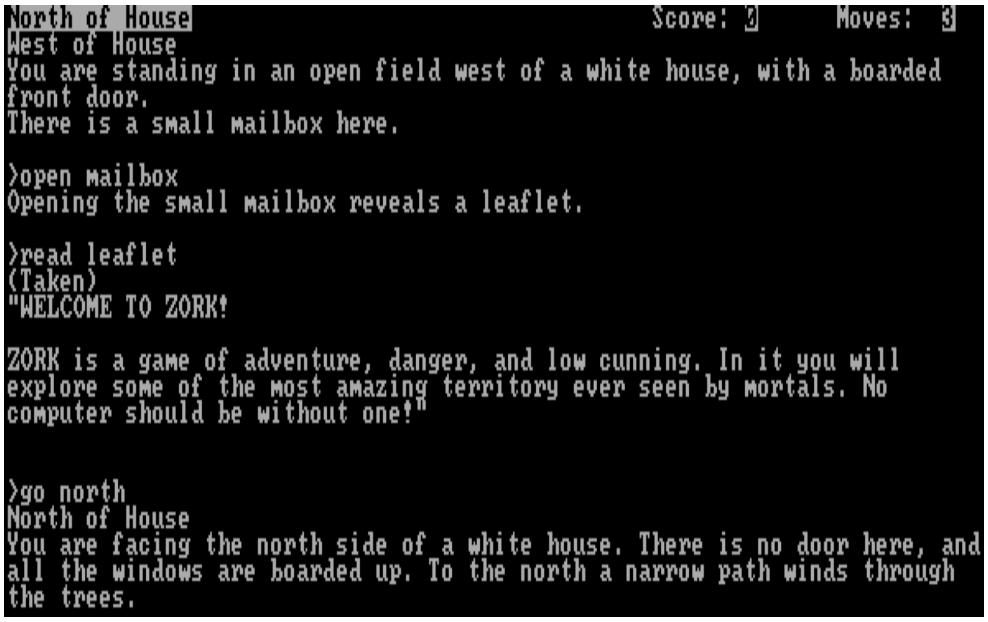


Figure 1.1: An excerpt from *Zork1*, a typical text-based adventure game.

Building on recent work in this field, I posit that interactive narratives should be the environments of choice for such language-oriented tasks. **Interactive Narratives**, in general, is an umbrella term, that refers to any form of digital interactive experience in which users create or influence a dramatic storyline through their actions (Riedl and Bulitko 2013)—i.e. the overall story progression in the game is not pre-determined and is directly influenced by a player's choices. For the purposes of this work, I consider one particular type of interactive narrative: parser-based interactive fiction (or text-adventure) games—though I note that other forms of interactive narrative, including those with visual components, provide closely related challenges.

Figure 1.1 showcases *Zork* (Anderson *et al.* 1979), one of the earliest and most influential text-based interactive narrative. These games are simulations in which an agent interacts with the world through natural language—"perceiving", "acting upon", and "talking to" the world using textual descriptions, commands, and dialogue. The simulations are *partially observable*, meaning that the agent never has access to the true underlying world state and has to reason about how to act in the world based only on potentially the incomplete textual observations of its immediate surroundings. They provide tractable,

situated environments in which to explore highly complex interactive grounded language learning without the complications that arise when modeling physical motor control and vision—situations that voice assistants such as Siri or Alexa might find themselves in when improvising responses. These games are usually structured as puzzles or quests with long-term dependencies in which a player must complete a sequence of actions and/or dialogues to succeed. This in turn requires navigation and interaction with hundreds of locations, characters, and objects. The interactive narrative community is one of the oldest gaming communities and game developers in this genre are quite creative. Put these two things together and we get very large, complex worlds that contain a multitude of puzzles and quests to solve across many different genres—everything from slice of life simulators where the player cooks a recipe in their home to Lovecraftian horror mysteries. The complexity and diversity of topics enable us to build and test agents that go an extra step towards modeling the difficulty of situated human language communication.

As the excerpt of the text-game in Figure 1.1 shows, humans bring competencies in natural language understanding, commonsense reasoning, and deduction to bear in order to infer the context and objectives of a game. Beyond games, real-world applications such as voice-activated personal assistants can also benefit from advances in these capabilities at the intersection of natural language understanding, natural language generation, and sequential decision making. These real world applications require the ability to reason with ungrounded natural language (unlike multimodal environments that provide visual grounding for language) and interactive narratives provide an excellent suite of environments to tackle these challenges.

1.1 Challenges of Operating in Interactive Environments

Interactive narratives exist at the intersection of natural language processing, storytelling, and sequential decision making. Like many NLP tasks, they require natural language understanding, but unlike most NLP tasks, Interactive narratives are sequential decision mak-

Figure 1.2: A map of *Zork1* by artist *ion_bond*.

ing problems in which actions change the subsequent world states of the game and choices made early in a game may have long term effects on the eventual endings. Reinforcement Learning (Sutton and Barto 1998) studies sequential decision making problems and has shown promise in vision-based (Jaderberg *et al.* 2016) and control-based (OpenAI *et al.* 2018) environments, but has less commonly been applied in the context of language-based tasks. Text-based games thus pose a different set of challenges than traditional video games such as *StarCraft*. Their puzzle-like structure coupled with a partially observable state space and sparse rewards require a greater understanding of previous context to enable more effective exploration—an implicit *long-term dependency* problem not often found in other domains that agents must overcome.

1.1.1 Knowledge Representation

Interactive narratives span many distinct locations, each with unique descriptions, objects, and characters. An example of a world of a interactive fiction game can be seen in Figure 1.2. Players move between locations by issuing navigational commands like *go West*.

This, in conjunction with the inherent *partial observability* of interactive narratives, gives rise to the **Textual-SLAM** problem, a textual variant of Simultaneous Localization and Mapping (SLAM) (Thrun *et al.* 2005) problem of constructing a map while navigating a new environment. In particular, because connectivity between locations is not necessarily Euclidean, agents need to detect when a navigational action has succeeded or failed and whether the location reached was previously seen or new. Beyond location connectivity, it's also helpful to keep track of the objects present at each location, with the understanding that objects can be nested inside of other objects, such as food in a refrigerator or a sword in a chest.

Due to the large number of locations in many games, humans often create structured memory aids such as maps to navigate efficiently and avoid getting lost. The creation of such memory aids has been shown to be critical in helping automated learning agents operate in these textual worlds (Ammanabrolu and Riedl 2019b; Murugesan *et al.* 2020; Adhikari *et al.* 2020; Ammanabrolu and Hausknecht 2020).

1.1.2 Commonsense Reasoning

Many real-world activities can be thought of as a sequence of sub-goals in a partially observable environment. These activities—getting ready to go to work, for example—are considered trivial for humans because of *commonsense knowledge*. Commonsense knowledge is defined as a set of facts, beliefs, and procedures shared among many people in the same society or culture. However, to an agent learning purely by interacting with the environment, even simple tasks can require considerable trial-and-error. I hypothesize that access to commonsense knowledge can enable an agent to more quickly converge on a policy that completes common, everyday tasks. I further hypothesize that commonsense knowledge can allow the agent to infer the presence of elements in the world when observations are noisy or fail.

Text-games cover a wide variety of genres, as mentioned earlier this ranges from slice

of life simulators where the player makes a recipe in their home to mysteries and fairy tales. This enables us to explore the question of how to adapt to domain-specific knowledge which may contradict everyday commonsense. Take for example, an agent that knows that knows how to cut vegetables with a knife. When placed in an environment without a knife, it must adapt its cooking knowledge to account for this in order to still construct the recipe successfully.

In order to effectively convey the core narrative or puzzle, text-adventure games make ample use of prior commonsense and thematic knowledge. An everyday example could be something as mundane as the fact that an axe can be used to cut wood, or that swords are weapons. Different genres also have specific knowledge attached to them that wouldn't normally be found in mundane settings, e.g. in a horror or fantasy game, we know that a coffin is likely to contain a vampire or other undead monster or that kings are royalty and must be treated respectfully. When a human enters a particular domain, they already possess priors regarding the specific knowledge relevant to the situations likely to be encountered—this is thematic commonsense knowledge that a learning agent must acquire to ensure successful interactions.

This is closely related to the problem of *transfer*, the problem of acquiring and adapting these priors in novel environments through interaction. In this sense, we can think of commonsense knowledge as priors regarding environment dynamics. This problem space can be explored using text-based games. What commonsense can be transferred between two different environments, for example, a horror game and a mundane slice of life game? How do you unlearn, or choose not to apply, a piece of commonsense that no longer fits with the current world. What if the perceived environment dynamics change in novel ways? E.g. some vampires actually love garlic instead of being allergic to them or you suddenly find out that bread can be made without yeast and is known as sourdough—whole new categories of recipes are now possible.

1.1.3 Acting in Combinatorially-sized State-Action Spaces

Interactive narratives require the agent to operate in the combinatorial action space of natural language. To realize how difficult a game such as *Zork1* is for standard reinforcement learning agents, we need to first understand how large this space really is. In order to solve a popular IF game such as *Zork1* it's necessary to generate actions consisting of up to five-words from a relatively modest vocabulary of 697 words recognized by Zork's parser. Even this modestly sized vocabulary leads to $\mathcal{O}(697^5) = 1.64 \times 10^{14}$ possible actions at every step—a dauntingly-large *combinatorially-sized action space* for a learning agent to explore. In comparison, board games such as chess and Go or Atari video games have branching factors of the order of $\mathcal{O}(10^2)$.

1.1.4 Exploration

Most text-adventure games are structured as quests with high branching factors in which players must solve a sequence of puzzles to advance the story and gain score—i.e. there are usually multiple ways to finish a quest. To solve these puzzles, players have freedom to a explore both new areas and previously unlocked areas of the game, collect clues, and acquire tools needed to solve the next puzzle and unlock the next portion of the game. From a Reinforcement Learning perspective, these puzzles can be viewed as bottlenecks that act as partitions between different regions of the state space. Whereas the multiple pathways to completion through puzzles may intuitively seem to make the problem easier, the opposite is true. The bottlenecks set up a situation where agents get stuck because they do not see the right action sequence enough times to be sufficiently reinforced. I contend that existing Reinforcement Learning agents are unaware of such latent structure and are thus poorly equipped for solving these types of problems.

Overcoming bottlenecks is not as simple as selecting the correct action from the bottleneck state. Most bottlenecks have long-range dependencies that must first be satisfied: *Zork1* for instance features a bottleneck in which the agent must pass through the unlit *Cel-*

lar where a monster known as a Grue lurks, ready to eat unsuspecting players who enter without a light source. To pass this bottleneck the player must have previously acquired and lit the lantern. Reaching the *Cellar* without acquiring the lantern results in the player reaching an *unwinnable state*—the player is unable to go back and acquire a lantern but also cannot progress further without a way to combat the darkness. Other bottlenecks don't rely on inventory items and instead require the player to have satisfied an external condition such as visiting the reservoir control to drain water from a submerged room before being able to visit it. In both cases, the actions that fulfill dependencies of the bottleneck, e.g. acquiring the lantern or draining the room, are not rewarded by the game. Thus agents must correctly satisfy all *latent* dependencies, most of which are unrewarded, then take the right action from the correct location to overcome such bottlenecks. Consequently, most existing agents—regardless of whether they use a reduced action space (Zahavy *et al.* 2018; Yuan *et al.* 2018; Yin and May 2019a) or the full space (Ammanabrolu and Hausknecht 2020; Hausknecht *et al.* 2020)—have failed to consistently clear these bottlenecks. It is only recently that works have begun explicitly accounting for and surpassing such bottlenecks—using a reduced action space and Monte-Carlo Planning (Jang *et al.* 2021) and full action space and intrinsic motivation-based structured exploration (Ammanabrolu *et al.* 2020d).

While problems relating to long-term dependencies and sparse rewards are not unique to text games alone, they are significantly complicated in this domain due to agents having to simultaneously handle all the other challenges as well. As a result, even existing algorithms designed for the current test beds of choice for these issues such as GoExplore for the Atari game Montezuma's Revenge (Ecoffet *et al.* 2021) face difficulties in overcoming bottlenecks in this domain (Ammanabrolu *et al.* 2020d; Madotto *et al.* 2020).

1.1.5 Simultaneously Learning to Act and Speak

Some text-games extend the previous challenges even further by requiring agents to engage in dialogue to progress in a task, increasing the space of possibilities exponentially

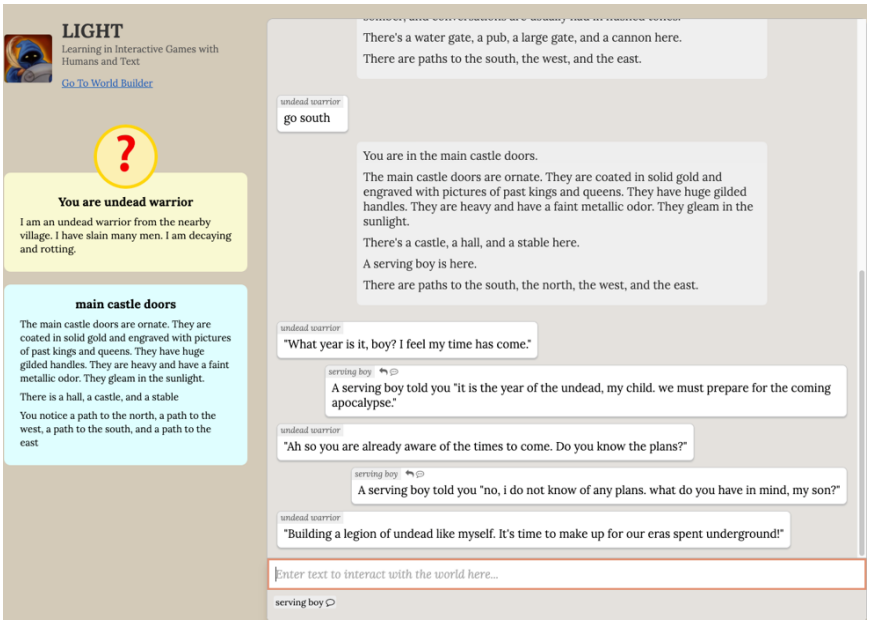


Figure 1.3: The *LIGHT* (Urbanek *et al.* 2019) environment.

and bringing text environments closer to real-world situations. An example of such an environment—designed explicitly as a research platform— is the large-scale crowdsourced fantasy text-adventure game *LIGHT* (Urbanek *et al.* 2019), seen in Figure 1.3, where characters can act and talk while interacting with other characters. It consists of of a set of locations, characters, and objects leading to rich textual worlds in addition to quests demonstrations of humans playing these quests providing natural language descriptions in varying levels of abstraction of motivations for a given character in a particular setting.

On top of the other text-game related challenges, the primary core challenge for the agent here is the recognition that dialogue can also be used to change the environment. With dialogue, an agent can now learn to instruct or convince other characters in the world to achieve the goal for it—e.g. convince the pirate through dialogue to give you their treasure instead of just stealing it yourself. The agent needs to learn to balance both its ability to speak as well as act in order to effectively achieve its goals (Ammanabrolu *et al.* 2021).

1.2 Challenges of Generating Interactive Environments

A key consideration in modeling communication through a general purpose interactive narrative solver is that an agent trained to solve these games is limited by the scenarios described in them. Although the range of scenarios is vast, this brings about the question of what the agent is actually capable of understanding even if it has learned to solve all the puzzles in a particular game. Deep (reinforcement) learning systems tend to learn to generalize from the head of any particular data distribution, the "common" scenarios, and memorize the tail, the rarely seen cases. I contend that a potential way of testing an AI system's understanding of a domain is to use the knowledge it has gained in a novel way and to create more instances of that domain. We can view this as storytelling—long considered to be one of our most natural forms of communication (Boyd 2018).

From the perspective of interactive narratives, this involves automatically creating such games—the flip side of the problem of creating agents that operate in these environments—and requires *anticipating* how people will interact with these environments and conforming to such expected commonsense norms to make a creative and engaging experience. Automated generation of text-adventure games can broadly be split into two considerations: (1) the structure of the world, including the layout of rooms, textual description of rooms, objects, and non-player characters; and (2) the quest, consisting of the partial ordering of activities that the player must engage in to make progress toward the end of the game.

1.2.1 Quest Generation

The core experience in an interactive narrative revolves the quest, consisting of the partial ordering of activities that an agent must engage in to make progress toward the end of the game. *Quest generation* requires narrative intelligence and commonsense knowledge as a quest must maintain coherence throughout while progressing towards a goal (Ammanabrolu *et al.* 2020a). Each step of the quest follows logically from the preceding steps

much like the steps of a cooking recipe. A restaurant cannot serve a batch of cookies without first gathering ingredients, preparing cooking instruments, mixing ingredients, etc. in a particular sequence. Any generated quest that doesn't follow such an ordering will appear random or nonsensical to a human, betraying the AI's lack of commonsense understanding.

1.2.2 World Generation

Maintaining quest coherence also means following the constraints of the given game world. The quest has to fit within the confines of the world in terms of both genre and given affordances—e.g. using magic in a fantasy world, placing kitchens next to living rooms in mundane worlds, etc. This gives rise to the concept of *world generation*, the second half of the automated game generation problem. This refers to generating the structure of the world, including the layout of rooms, textual description of rooms, objects, and characters—setting the boundaries for how an agent is allowed to interact with the world (Ammanabrolu *et al.* 2020b). Similarly to quests, a world violating thematically relevant commonsense structuring rules will appear random to humans, providing us with a metric to measure an AI system's understanding.

1.3 Thesis Statement

This bring us to the thesis statement and how I propose to tackle all of these problems. **Structured contextualization, in the form of knowledge graphs, improves situated natural language generation and understanding in interactive environments as evaluated by (1) the ability to operate in textual worlds, and (2) the perceived coherence and creativity of procedurally generated language-based environments.**

A core component of this thesis is generating and understanding language in situated—or grounded—interactive environments. This requires an understanding of context, the descriptions of the world must be interpreted and language generated accordingly. Structuring this context in the form of a knowledge graph aid in each of the challenges discussed

so far. This work is structured into three parts:

Part 1: Operating in Textual Worlds. In this part I focus on creating agents that can learn to act and speak in interactive narratives, highlighting the effectiveness of knowledge graphs. First, I show that graphs provide knowledge representations containing a persistent memory—letting us overcome the challenge of partially observable state spaces (Chapter 3). Combinatorially-large natural language action spaces can be constrained based on information contained withing an agent's knowledge graph (Chapter 4). Chapter 5 extends our ability to explore combinatorially-sized spaces by exploiting the latent, underlying dependency graph structure of these POMDPs via knowledge graph-based intrinsic motivation. These graphs let us seed agents with external commonsense knowledge as well as to transfer prior commonsense and thematic knowledge that they have learned (Chapter 6). Chapter 7 builds on these challenges by looking the question of how to balance dual act-speech spaces by learning to simultaneously perform goal-driven, situated dialogue while also acting. The challenges measure an agent's ability to operate in a interactive textual environment and so are evaluated based on game-playing ability—i.e. how well an agent can complete a given text-adventure game.

Part 2: Generating Textual Worlds. On the game generation side of things, we factorize the problem into the problems of world and quest generation. Graph representations let us structure and develop textual worlds that align with thematic and everyday commonsense priors (Chapter 8) and let us ground objectives—or quests—into these worlds (Chapter 9). These challenges in game generation are evaluated by humans in terms of how creative and coherent they perceive the generated environments to be.

Part 3: Putting it all together. Chapter 10 ties both the game playing and game generation lines of research together by proposing a method to train agents to act and speak via natural language using curriculums of procedurally generated textual environments.

CHAPTER 2
BACKGROUND AND PRIOR WORK

This chapter first provides a background on the formal definition of text games as applied to interactive, learning setting and the two primary frameworks used through the rest of this dissertation. I then sketch related work in this area both in the realms of game-playing and game-generation.

2.1 Frameworks

2.1.1 Background

Formally, text-adventure games can be defined as Partially-Observable Markov Decision Processes (Hausknecht *et al.* 2020; Côté *et al.* 2018). A game can be represented as a 7-tuple of $\langle S, T, A, \Omega, O, R, \gamma \rangle$ representing the set of environment states, *mostly deterministic conditional transition probabilities between states*, the vocabulary or words used to compose text commands, observations returned by the game, observation conditional probabilities, reward function, and the discount factor respectively. At every step, an agent receives an observation from the environment, then chooses an action to perform and receives an updated observation from the game engine.

I have also aided in the development of the primary open-source platforms and baseline benchmarks in this field: *Jericho* a learning environment for human-made interactive narrative games; and *LIGHT* a large-scale crowdsourced multi-user text-game for studying situated dialogue—each resulting in hundreds of stars and forks on GitHub and dozens of agents. These are used through the rest of this dissertation.

2.1.2 Jericho

Jericho is an open-source[1] Python-based interactive narrative environment, which provides an OpenAI-Gym-like interface (Brockman *et al.* 2016) for learning agents to connect with interactive narrative games. Jericho is intended for reinforcement learning agents, but also supports the ability to load and save game states, enabling planning algorithms Monte-Carlo Tree Search (Coulom 2007) as well as reinforcement learning approaches that rely on the ability to restore state such as Backplay (Resnick *et al.* 2018) and GoExplore (Ecoffet *et al.* 2021). Jericho additionally provides the option to seed the game's random number generator for replicability.[2]

Jericho supports a set of human-made interactive narrative games that cover a variety of genres: dungeon crawl, Sci-Fi, mystery, comedy, and horror. Games were selected from classic Infocom titles such as *Zork* and *Hitchhiker's Guide to the Galaxy*, as well as newer, community-created titles like *Anchorhead* and *Afflicted*. Supported games use a point-based scoring system, which serves as the agent's reward. Beyond the set of supported games, unsupported games may be played through Jericho, without the support of score detection, move counts, or world-change detection. There exists a large collection of over a thousand unsupported games[3], which may be useful for unsupervised pretraining or intrinsic motivation.

Template-Based Action Generation I introduce a novel template-based action space in which the agent first chooses an action template (e.g. *put _ in _*) and then fills in the blanks using words from the parser's vocabulary. Notationally, I employ $u \Leftarrow w_1, w_2$ to denote the filling of template u with vocabulary words w_1, w_2. Jericho provides the capability to extract game-specific vocabulary and action templates. These templates contain up to two blanks, so a typical game with 200 templates and a 700 word vocabulary yields an action

[1] Jericho is available at https://github.com/Microsoft/jericho.

[2] Most interactive narrative games are deterministic environments. Notable exceptions include Anchorhead and Zork1.

[3] https://github.com/BYU-PCCL/z-machine-games

space of $\mathcal{O}(\mathcal{T}\mathcal{V}^2) \approx 98$ million, three orders of magnitude smaller than the 240-billion space of 4-word actions using vocabularly alone.

World Object Tree The world object tree[4] is a semi-interpretable latent representation of game state used to codify the relationship between the objects and locations that populate the game world. Each object in the tree has a *parent*, *child*, and *sibling*. Relationships between objects are used to encode posession: a location object contains children corresponding to the items present at that location. Similarly, the player object has the player's current location as a parent and inventory items as children. Possible applications of the object tree include ground-truth identification of player location, ground-truth detection of the objects present at the player's location, and world-change-detection.

Identifying Valid Actions *Valid actions* are actions recognized by the game's parser that cause changes in the game state. When playing new games, identifying valid actions is one of the primary difficulties encountered by humans and agents alike. Jericho has the facility to detect valid actions by executing a candidate action and looking for resulting changes to the world-object-tree. However, since some changes in game state are reflected only in global variables, it's rare but possible to experience false negatives. In order to identify all the valid actions in a given state, Jericho uses the following procedure:

Handicaps In summary, to ease the difficulty of interactive narrative games, Jericho optionally provides the following handicaps: 1) Fixed random seed to enforce determinism. 2) Use of Load, Save functionality. 3) Use of game-specific templates $\mathcal{U}$ and vocabulary $\mathcal{V}$. 4) Use of world object tree as an auxillary state representation or method for detecting player location and objects. 5) Use of world-change-detection to identify valid actions. For reproducibilty, I report the handicaps used by all algorithms in this chapter and encourage future work to do the same.

[4]More on game trees https://inform-fiction.org/zmachine/standards/z1point1/index.html.

Algorithm 1 Procedure for Identifying Valid Actions

$\mathcal{E} \leftarrow$ Jericho environment
$\mathcal{T} \leftarrow$ Set of action templates
$o \leftarrow$ Textual observation
$\mathcal{P} \leftarrow \{p_1 \dots p_n\}$ Interactive objects identified with noun-phrase extraction or world object tree.
$Y \leftarrow \varnothing$ List of valid actions
$s \leftarrow \mathcal{E}.save()$ – Save current game state
for template $u \in \mathcal{T}$ **do**
    for all combinations $p_1, p_2 \in \mathcal{P}$ **do**
        Action $a \leftarrow u \Leftarrow p_1, p_2$
        if $\mathcal{E}.world\_changed(\mathcal{E}.step(a))$ **then**
            $Y \leftarrow Y \cup a$
            $\mathcal{E}.load(s)$ – Restore saved game state
return Y

2.1.3 LIGHT

This section first provides a brief overview of the LIGHT game environment, followed by descriptions of the LIGHT-Quests and ATOMIC-LIGHT datasets used in this chapter.

The LIGHT game environment is a multi-user fantasy text-adventure game consisting of a rich, diverse set of characters, locations, and objects (1775 characters, 663 locations, and 3462 objects). Characters are able to perform templated actions to interact with both objects and characters, and can speak to other characters through free form text. Actions in text games generally consist of verb phrases (VP) followed optionally by prepositional phrases (VP PP). For example, *get OBJ, put OBJ, give OBJ to CHAR*, etc.. There are 13 types of allowed verbs in LIGHT. These actions change the state of the world which is expressed to the player in the form of text descriptions.

Figures 2.1, 2.2, and 2.3 summarize the data that I collected for LIGHT-Quests. Data is collected via crowdsourcing in two phases, first the quests then demonstration of humans playing them. During the first phase, crowdworkers were given a setting, i.e. situated in a world, in addition to a character and its corresponding persona and asked to describe in free form text what potential motivations or goals could be for that character in the given world. The kind of information given to the crowdworkers is seen in Figure 2.1. Simultaneously,

Setting	You are in the Dangerous Precipice. The dangerous precipice overlooks the valley below. The ground slopes down to the edge here. Dirt crumbles down to the edge of the cliff. There's a dragon crescent, a knight's armor, a golden dragon egg, and a knight's fighting gear here. A knight is here.You are carrying nothing.
Partner:	Knight.
Persona	I am a knight. I come from a lower-ranking noble family. I serve under the king, as my father did before me. In times of war, I fight on horseback.
Carrying	knight's armor, golden dragon egg, knight's fighting gear
Self:	A dragon.
Persona	I am a dragon living in the mountains. I enjoy hoarding treasure. I terrorize the local populace for fun.
Carrying	Nothing.

Figure 2.1: Setting and character information for both self and partner characters as taken from LIGHT.

Motivations:		**Timeline:**	
Short	I need to recover the dragon egg that was stolen and punish the knight.	-4 hours	go to dangerous precipice
		-15 min	get knights armor from knight
		-10 min	get golden dragon egg
Mid	I need to return the golden dragon egg to my treasure hoard.	Now	hit knight
		+5 min	put dragon egg on back
		+15 min	eat the knight
Long	I need to build the largest hoard ever attained by any one dragon.	+2 hours	go to the mountains

Figure 2.2: Motivations with different levels of abstractions and corresponding sequence of timeline actions in chronological order for the self character in LIGHT-Quests. There are 7486 quests in total.

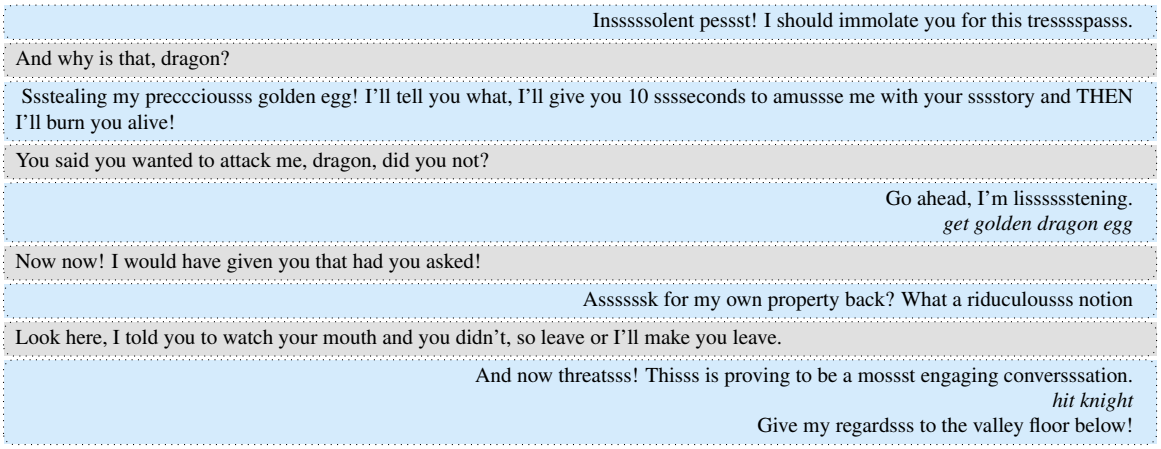


Figure 2.3: Example of a demonstration of a human (blue shaded) completing the above quest while role-playing as the self character with a partner agent (grey shaded). There are 2111 such human demonstrations of average sequence length 12.92, consisting of 22672 dialogues in total.

they were also asked to provide a sequence of seven timeline actions—one action that needs to be completed *now* and three before and after at various user-defined intervals—for how the character might go about achieving these motivations.

Given the information in Figure 2.1, the crowdworkers completed the above outlined tasks and produce data as seen in Figure 2.2. Motivations come in three levels of abstraction—short, mid, and long—corresponding to differing amounts of the timeline. For example, the

short motivation is always guaranteed to correspond most closely to the *now* position on the timeline. Action annotation is pre-constrained based on the classes of verbs available within LIGHT. The rest of the action is completed as free form text as it may contain novel entities introduced in the motivations. There are 5982 training, 756 validation, and 748 test quests. Further details regarding the exact data collection process and details of LIGHT-Quests are found in Appendix B.4.

After collecting motivation and timelines for the quests, I deployed a two-player version of the LIGHT game, letting players attempt the quests for themselves in order to collect human demonstrations. Figure 2.3 shows an example human expert demonstration of a quest. Players were given a character, setting, motivation, and a partner agent and left to freely act in the world and talk to the partner in pursuit of their motivations. The partner agent is a fixed poly-encoder transformer model (Humeau *et al.* 2020) trained on the original LIGHT data as well as other human interactions derived via the deployed game—using 111k utterances in total. Players first receive a role-playing score on a scale of 1-5 through a Dungeon Master (DM), a learned model that ranks how likely their utterances are given the current context. Once they have accumulated a score reaching a certain threshold, they are allowed to perform actions. I employ this gamification mechanism to encourage players to role-play their character persona and its motivations, leading to improved user experience and data quality (Horsfall and Oikonomou 2011). They are then given further reward if the actions they perform sequentially match those on the timeline for the given quest. The game ends after a maximum of six turns of dialogue per agent, i.e. twelve in total. The average sequence of a human demonstration is 12.92, with an average action sequence length of 2.18 and dialogue of 10.74. There are 1800 training, 100 validation, and 211 test human expert demonstrations after the data was filtered. Additional details and examples are found in Appendix B.4.1.

2.2 Related Work on Language-Based Agents

Currently, three primary open-source platforms and baseline benchmarks have been developed so far to help measure progress in this field: *Jericho* (Hausknecht *et al.* 2020)[5] a learning environment for human-made interactive narrative games; *TextWorld* (Côté *et al.* 2018)[6] a framework for procedural generation in text-games; and *LIGHT* (Urbanek *et al.* 2019)[7] a large-scale crowdsourced multi-user text-game for studying situated dialogue. Further extensions and adaptation to some of these benchmarks have been proposed for use in neighboring domains such as vision-and-language navigation (Shridhar *et al.* 2021), commonsense reasoning (Murugesan *et al.* 2021), and procedural text understanding (Tamari *et al.* 2021). In this dissertation, we focus on the first three mentioned.

Text Game Playing. In contrast to the *parser-based games* studied in this dissertation, *choice-based games* provide a list of possible actions at each step, so learning agents must only choose between the candidates. The DRRN algorithm for choice-based games (He *et al.* 2016a; Zelinka 2018) estimates Q-Values for a particular action from a particular state. This network is evaluated once for each possible action, and the action with the maximum Q-Value is selected. While this approach is effective for choice-based games which have only a handful of candidate actions at each step, it is difficult to scale to parser-based games where the action space is vastly larger.

In terms of *parser-based games*, such as the ones examined in this dissertation, several approaches have been investigated: LSTM-DQN (Narasimhan *et al.* 2015), considers *verb-noun* actions up to two-words in length. Separate Q-Value estimates are produced for each possible verb and object, and the action consists of pairing the maximally valued verb combined with the maximally valued object. LSTM-DQN was demonstrated to work on two small-scale domains, but human-made games, such as those studied in this chapter, represent a significant increase in both complexity and vocabulary. This bifurcation of

[5] https://github.com/microsoft/jericho
[6] https://github.com/microsoft/textworld
[7] https://parl.ai/projects/light

value estimates allows the agent to reason about exponentially fewer actions, at the risk of selecting poorly matched verb-object pairs.

One approach to affordance extraction (Fulda *et al.* 2017) identified a vector in word2vec (Mikolov *et al.* 2013) space that encodes affordant behavior. When applied to the noun *sword*, this vector produces affordant verbs such as *vanquish*, *impale*, *duel*, and *battle*. The authors use this method to prioritize verbs for a Q-Learning agent to pair with in-game objects.

An alternative strategy has been to reduce the combinatorial action space of parser-based games into a discrete space containing the minimum set of actions required to finish the game. This approach requires a walkthrough or expert demonstration in order to define the space of minimal actions, which limits its applicability to new and unseen games. Following this approach, Zahavy *et al.* (2018) employ this strategy with their action-elimination network, a classifier that predicts which predefined actions will not effect any world change or be recognized by the parser. Masking these invalid actions, the learning agent subsequently evaluates the set of remaining valid actions and picks the one with the highest predicted Q-Value.

The TextWorld framework (Côté *et al.* 2018) supports procedural generation of parser-based interactive narrative games, allowing complexity and content of the generated games to be scaled to the difficulty needed for research. TextWorld domains have already proven suitable for reinforcement learning agents (Yuan *et al.* 2018) which were shown to be capable of learning on a set of environments and then generalizing to unseen ones at test time. Recently, Yuan *et al.* (2019) proposed QAit, a set of question answering tasks based on games generated using TextWorld. QAit focuses on helping agents to learn procedural knowledge in an information-seeking fashion, it also introduces the practice of generating unlimited training games on the fly. With the ability to scale the difficulty of domains, TextWorld enables creating a curriculum of learning tasks and helping agents eventually scale to human-made games.

In the case of human-made text games, however, knowledge graphs—not directly provided by existing text game learning frameworks—have been shown to be superior state representations when compared to just the textual observations by themselves. They aid in the challenges of partial observability/knowledge representation (Ammanabrolu and Riedl 2019b; Adhikari *et al.* 2020; Sautier *et al.* 2020), combinatorial state-action spaces (Ammanabrolu and Hausknecht 2020; Ammanabrolu *et al.* 2020d), and commonsense reasoning (Ammanabrolu and Riedl 2019b; Murugesan *et al.* 2020, 2021; Dambekodi *et al.* 2020). These rest of the chapters in this dissertation will go explicitly into detail on the uses of knowledge graphs in text games.

Transfer. Work in transfer in reinforcement learning has explored the idea of transferring skills (Konidaris and Barto 2007; Konidaris *et al.* 2012) or transferring value functions/policies (Liu and Stone 2006). Other approaches attempt transfer in model-based reinforcement learning (Taylor *et al.* 2008; Nguyen *et al.* 2012; Gasic *et al.* 2013; Wang *et al.* 2015; Joshi and Chowdhary 2018), though traditional approaches here rely heavily on hand crafting state-action mappings across domains. Narasimhan *et al.* (2017) learn to play games by predicting mappings across domains using a both deep Q-networks and value iteration networks, finding that that grounding the game state using natural language descriptions of the game itself aids significantly in transferring useful knowledge between domains.

In transfer for deep reinforcement learning, Parisotto *et al.* (2016) propose the Actor-Mimic network which learns from expert policies for a source task using policy distillation and then initializes the network for a target task using these parameters. Yin H. and Pan (2017) also use policy distillation, using task specific features as inputs to a multi-task policy network and use a hierarchical experience sampling method to train this multi-task network. Similarly, Rusu *et al.* (2016) attempt to transfer parameters by using frozen parameters trained on source tasks to help learn a new set of parameters on target tasks. Rajendran *et al.* (2017) attempt something similar but use attention networks to transfer

expert policies between tasks. These works, however, do not study the requirements for enabling efficient transfer for tasks rooted in natural language, nor do they explore the use of knowledge graphs as a state representation.

Exploration strategies. Jain *et al.* (2019) extend consistent Q-learning (Bellemare *et al.* 2016) to text-games, focusing on taking into account historical context. In terms of exploration strategies, Yuan *et al.* (2018) detail how counting the number of unique states visited improves generalization in unseen games. Madotto *et al.* (2020) use the GoExplore (Ecoffet *et al.* 2021) to specifically explore text games using valid action handicaps. Similarly, Jang *et al.* (2021) use valid action handicaps and linguistic priors to selectively perform rollouts using Monte Carlo Tree Search.

Curriculum Learning. Curricula in reinforcement learning have traditionally been used to set goals of steadily increasing difficulty for an agent (Bengio *et al.* 2009; Schmidhuber 2013). The difficulty of these curricula are generally measured difficulty via proxy of agent performance (Narvekar *et al.* 2020). Given this measure most methods either choose to adversarially set steadily goals of increasing difficulty (Sukhbaatar *et al.* 2018; Racaniere *et al.* 2019; Campero *et al.* 2021) or to maximize learning performance based on environment instances an agent finds difficult historically (Graves *et al.* 2017; Portelas *et al.* 2020). While we were inspired by these works, they all focus on searching for goals for agents which can be difficult to scale to complex tasks such our own natural language motivation-based goals.

Goal oriented Dialogue This form of dialogue has traditionally been closely related to specific tasks useful in the context of personal assistants with dialogue interfaces (Henderson *et al.* 2014; El Asri *et al.* 2017). RL has been studied for such tasks, usually to improve dialogue state management (Singh *et al.* 2000; Pietquin *et al.* 2011; Fatemi *et al.* 2016) and to improve response quality (Li *et al.* 2016). In particular, the negotiation tasks of Yarats and Lewis (2017) and Lewis *et al.* (2017), where two agents are trying to convince each other to perform certain actions, are related to the tasks in LIGHT-Quests. These works all

lack environment grounding and the notion of diverse agent motivations.

Commonsense reasoning in language. Trabasso and Broek (1985) as well as Graesser *et al.* (1991) introduce psychological theories relating commonsense reasoning with causality in natural language stories, wherein what is regarded as commonsense is based on the "why" and "how" of the activities a certain character needs to perform to reach goals consistent with their motivations. Works such as Bosselut *et al.* (2019) and Guan *et al.* (2020) focus on pre-training transformer-based language learning systems with large-scale commonsense knowledge graphs such as ATOMIC (Sap *et al.* 2019) and ConceptNet (Speer and Havasi 2012) for use in knowledge graph completion and story ending generation respectively. Murugesan *et al.* (2020), Ammanabrolu *et al.* (2020d), Fulda *et al.* (2017), Dambekodi *et al.* (2020), and Ammanabrolu and Riedl (2019a) look at commonsense reasoning in interactive environments, with the former focusing on affordance extraction using word embeddings and the latter three on transferring text-game playing skills via pre-training using question-answering and large-scale knowledge graphs.

Language-informed reinforcement learning. Luketina *et al.* (2019) provide an overview of RL informed by natural language. Of these works, the ones most related to ours are those falling into the category of instruction following—where an agent's tasks are defined by high level instructions describing desired policies and goals (MacMahon *et al.* 2006; Kollar *et al.* 2010). Visual and embodied agents using natural language instructions (Kolve *et al.* 2017; Bisk *et al.* 2016; Anderson *et al.* 2018) or in language-based action spaces (Das *et al.* 2017) utilize interactivity and environment grounding but have no notion of agent motivations, nor make any attempt to explicitly model commonsense reasoning.

World Models. World modeling via model-based reinforcement learning often serves to learn transition models of an environment to allow for simulation without actually interacting with the environment (Arulkumaran *et al.* 2017). Ha and Schmidhuber (2018) use Variational Autoencoders (VAEs) combined with recurrent neural networks to learn compressed state representations over time of visual reinforcement learning environments (Brock-

man *et al.* 2016). This model is then used to simulate an environment and learn a control policy in it. Other contemporary works attempt to also learn dynamics models using raw pixels in the context of games such as Atari (Oh *et al.* 2015; Kipf *et al.* 2020), and Super Mario Bros. (Guzdial *et al.* 2015) as well as 3D simulations (Kipf *et al.* 2020) and robotics (Watter *et al.* 2015; Wahlström *et al.* 2015). We note that in all of these works, in addition to the state space being raw pixels—the action space is fixed and orders of magnitude smaller than in text games.

2.3 History of Text-Game Generation

Outside of this, there has been some work on learning to create content in the context of interactive narrative. These systems mainly work to overcome a significant bottleneck in the form of the human authoring required to create such works. Permar and Magerko (2013) present a method of generating cognitive scripts required for freeform activities in the form of pretend play. Specifically, they use interactive narrative—a form of pretend play that requires a high level of improvisation and creativity and uses cognitive scripts acquired from multiple experience sources. They take existing cognitive scripts and blend them in the vein of more traditional conceptual blending (Veale *et al.* 2000; Zook *et al.* 2011) to create new blended scripts. Closely related is (Magerko and O'Neill 2012) who present a Co-Creative Cognitive Architecture (CoCoA), detailing the set of components that support the design of co-creative agents in the context of interactive narrative. These methods all follow singular cognitive models that do not learn to generate content automatically.

Li *et al.* (2012) present Scheherazade, a system which learns a plot graph based on stories written by crowd sourcing the task of writing short stories through Amazon Mechanical Turk. This plot graph contains details relevant for the coherence of the story and includes: plot events, temporal precedence, and mutual exclusion relations. The generated narrative contains events that can be executed from this plot graph by both players and non-player characters. Guzdial *et al.* (2015) introduce Scheherazade-interactive narrative, a system

that learns to generate choose-your-own-adventure style interactive fictions in which the player chooses from prescribed options. More recently, Martin *et al.* (2017) introduce a pipeline systems for improvisational storytelling agents capable of collaboratively creating stories. These agents first focus on creating a plot for the story and then expand that plot into natural language sentences.

Giannatos *et al.* (2011) use genetic algorithms to create new story plot points for an existing game of interactive fiction using an encoding known as a precedence-constraint graph. This graph gives the system information regarding the ordering of events that must happen in the game in order to advance. They demonstrate the workings of their system by generating additional content for the popular interactive fiction game *Anchorhead*, and show that this can be integrated into the original game.

The Game Forge system (Hartsook *et al.* 2011) also uses genetic algorithms to generate a game world and plot line for related type of game, a computer role playing game (CRPG). This work focuses on generating layouts and plot structures to create novel game worlds through with a fitness function based on a transition graph that encodes pre-built game requirements. Tamari *et al.* (2019) focus on extracting action graphs for sequential decision making problems such as material science experiments and turn them into text-adventure games. Fan *et al.* (2019) leverage LIGHT (Urbanek *et al.* 2019)—a crowdsourced dataset of fantasy text-adventure dialogues—to learn to generate interactive fiction worlds on the basis of locations, characters, and objects.

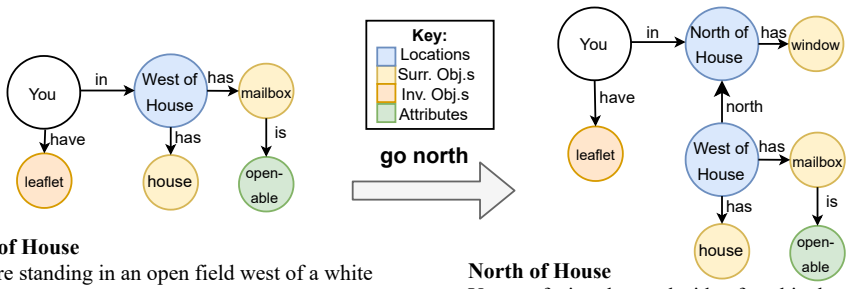
CHAPTER 3
MODELING PARTIALLY OBSERVABLE WORLDS

In this chapter, I focus on one of the core challenges faced by learning agents in these environments—as identified earlier—knowledge representation.

The **knowledge representation** challenge rises from the fact that interactive narratives span many distinct locations, each with unique descriptions, objects, and characters. Players move by issuing navigational commands, which can convey Euclidean space like *go West* or non-Euclidean span like *step into portal*, warping the agent to an entirely new section of the world. To cope with such challenges, humans often create structured memory aids such as hand drawn maps when attempting to play these games. A good knowledge representation can assist with long-term action dependencies that often arise in game quests (as well as real world environments). An example of a long-term dependency is a key being found in one location that opens a lock on a chest in an entirely different section of the map. For an agent to learn this relationship, it must be able to replicate the sequence of picking up the key and unlocking the chest while not being distracted by interstitial actions and states.

The knowledge representation challenges inherent to interactive narrative games give rise to the **Textual-SLAM** problem, a textual variant of Simultaneous Localization And Mapping (SLAM) (Thrun *et al.* 2005) problem of constructing a map by *inferring information* from one's surroundings while navigating a novel environment. As in humans, the creation of such world models or memory aids in agents—in the form of knowledge graphs—has been shown to be critical in helping automated learning agents operate in these textual worlds (Ammanabrolu and Riedl 2019b; Murugesan *et al.* 2020; Adhikari *et al.* 2020; Ammanabrolu and Hausknecht 2020).

I approach this problem of knowledge representation as a world modeling problem.



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 here.
Leaflet taken.
You are empty-handed
Prev act: take leaflet
Valid acts: go north, go south, go west, open mailbox

North of House
You are facing the north side of a white house. There is no door here, and all the windows are boarded up. To the north a narrow path winds through the trees.
You are carrying: a small leaflet
Prev act: go north
Valid acts: go north, go east, go west, drop leaflet

Figure 3.1: Two subsequent states in *Zork1* consisting of: textual observations, world knowledge graphs, valid actions, and actions taken.

World models, often in the form of probabilistic generative models, are used in conjunction with model-based reinforcement learning to improve a learning agent's ability to operate in various environments (Sutton and Barto 1998; Arulkumaran *et al.* 2017). They are inspired by human cognitive processes (Jancke 2000), with a key hypothesis being that the ability to predict how the world will change in response to one's actions will help you better plan what actions to take (Ha and Schmidhuber 2018). Evidence towards this hypothesis comes in the form of studies showing that simulating trajectories using internal learned models of the world improves sample efficiency in learning to operate in an environment (Ha and Schmidhuber 2018; Schrittwieser *et al.* 2019).

I show that a state representation in the form of a *knowledge graph* gives us the ability to not only map a textual world but also act more effectively in it. A knowledge graph captures the relationships between entities as a directed graph. The knowledge graph provides a persistent memory of the world over time and enables the agent to have a prior notion of what actions it should not take at a particular stage of the game.

3.1 Knowledge Graphs for POMDPs

Knowledge graphs have been demonstrated to improve natural language understanding in other domains outside of text adventure games. For example, Guan *et al.* (2018) use com-

monsense knowledge graphs such as *ConceptNet* (Speer and Havasi 2012) to significantly improve the ability of neural networks to predict the end of a story. They represent the graph in terms of a knowledge context vector using features from *ConceptNet* and graph attention (Veličković *et al.* 2018). The state representation that I have chosen as well as my method of action pruning builds on the strengths of existing approaches while simultaneously avoiding the shortcomings of ineffective exploration and lack of long-term context.

In my approach, my agent learns a knowledge graph, stored as a set of RDF triples, i.e. 3-tuples of $\langle subject, relation, object \rangle$. For example, from a phrase such as "There is an exit to the north" one can infer a *has* relation between the current location and the direction of the exit. The resultant knowledge graph gives the agent what essentially amounts to a mental map of the game world. The knowledge graph is updated after every agent action (see Figure 3.1). A special node—designated "you"—represents the agent and relations out of this node are updated after every action with the exception of relations denoting the agent's inventory. Other relations persist after each action.

3.1.1 JerichoWorld Dataset

In order to learn these knowledge graphs, I introduce and use the JerichoWorld Dataset.[1] It contains 24,198 mappings between rich natural language observations and: (1) knowledge graphs in the form of a set of tuples $\langle s, r, o \rangle$ (such that s is a subject, r is a relation, and o is an object) that reflect the world state in the form of a map; (2) a set of natural language actions that are guaranteed to cause a change in that particular world state. An example of the mapping between rich natural language observations and structured knowledge is illustrated in Figure 3.1. The training data is collected across 27 text games in multiple genres and contains a further 7,836 heldout instances over 9 additional games in the test set.

Each instance of the dataset takes the form of a tuple of the form $\langle S_t, A, S_{t+1}, R \rangle$ where

[1] https://github.com/JerichoWorld/JerichoWorld

S_t and S_{t+1} are two subsequent states with A being the action used to transition between states and R the observed reward. As mentioned earlier, each of the states in the tuple contain information regarding the observation $O_t \in S_t$, ground truth knowledge graph $G_t \in S_t$, and valid actions for that state $V_t \in S_t$. This data was collected by oracle agents, i.e. agents that can perfectly solve a game, exploring using a mix of an oracle and random policy to ensure high coverage of a game's state space. Game walkthroughs are texts describing the solutions to games, generally retrieved from the internet, but already part of the Jericho framework. Walkthroughs, however, only present one possible solution to a game and solve all the core puzzles required to complete a game with the maximum possible score.

To achieve greater coverage of the game's state space, my data collection agent stops off to explore by executing random valid actions for n steps before resetting to the walkthrough. One such collected state—a part of the full tuple mentioned—is detailed below.

The textual observations consist of descriptions of the location and inventory as well as the game engine response to the previous action performed. For example:

```
Game: ztuu
Location: Cultural Complex This imposing ante-room, the center of what was apparently the
    cultural center of the GUE, is adorned in the ghastly style of the GUE's "Grotesque
    Period." With leering gargoyles, cartoonish friezes depicting long-forgotten scenes of
     GUE history, and primitive statuary of pointy-headed personages unknown (perhaps very
    , very distant progenitors of the Flatheads), the place would have been best left
    undiscovered. North of here, a large hallway passes under the roughly hewn inscription
     "Convention Center." To the east, under a fifty-story triumphal arch, a passageway
    the size of a large city boulevard opens into the Royal Theater. A relatively small
    and unobtrusive sign (perhaps ten feet high) stands nearby. South, a smaller and more
    dignified (i.e. post-Dimwit) path leads into what is billed as the "Hall of Science."
    You can see a pair of razor-like gloves here.
Observation: You put on the razor-like gloves.
Inventory:
   You are carrying:
    a brass lantern (providing light)
    a pair of glasses
    four candy bars:
```

```
      a ZM$100000
      a Multi-Implementeers
      a Forever Gores
      a Baby Rune
    a cheaply-made sword
Prev Act: put on gloves
```

I further provide the set of objects that are found in both the agent's inventory and surroundings, including textual descriptions for each of the objects. Attributes for each of these objects are also included are acquired by decompiling the games, following (Ammanabrolu *et al.* 2020d). For example:

```
Inventory Objects:
   candy: Which do you mean, the ZM$100000, the Multi Implementeers, the Forever Gores or
       the Baby Rune?
   Implementeers: The profiles on the wrapper of this delicacy look more like Moe, Larry,
       and Curly than those of your favorite Implementeers (presumably, Marc, Mike, and
       David.)
   Forever/Gores: The wrapper of this bar pictures the Milky Way, but the stars are all
       blood red. Kids love them.
   sword: This is a cheaply made sword of no antiquity whatsoever. With regard to grues or
        other underworldly denizens, your weapon is as likely to engender laughter as fear
       .
   rune: The label is covered with mystical runes, the meanings of which elude you.
   glasses: The owner of these glasses had an indeterminate vision problem, because the
       lenses have both been crushed underfoot. The vision problem, of course, has been
       solved.
   lantern: The lantern, while of the cheapest construction, appears functional enough for
        the moment. Your best hope is that it stays that way. It looks like the lamp has
       gone through a few cycles of impact revitalization.
Inventory Attributes:
   glasses: clothing
   gloves: clothing
   sword: animate, equip
   lantern: animate, equip
Surrounding Objects:
   gargoyles: Unless you are inordinately masochistic, the less time spent examining the
       artwork, the better.
   east: You see nothing special about the east wall.
   tunnel: The tunnel leads west.
```

```
   gloves: The razor like gloves would be very attractive for an axe murderer. And they're
        just your size.
   south: You see nothing special about the south wall.
   sign: The sign indicates today's performance, which (in honor of the festivities in the
        Convention Center) is "A Massacre on 34th Street."
Surrounding Attributes:
   gloves: clothing
   tunnel: animate
   sign: animate
```

I further provide the ground truth knowledge graph representing the world state corresponding to these textual observations. The ground truth knowledge graph is a set of tuples $\langle s, r, o \rangle$ such that s is a subject, r is a relation, and o is an object. It reflects information on the current state such as objects and attributes and is extracted from the game engine by traversing the engine's internal representation and converting it to human readable form. Relations are defined on the basis of traversal operations in the game engine's internal representation, e.g. "in" and "have" signify parent-child ownership for locations and inventory respectively. For example:

```
Graph: [sign, in, Cultural Complex], [you, have, Forever Gores], [you, have, ZM$100000], [
    you, have, Baby Rune], [tunnel, in, Cultural Complex], [you, in, Cultural Complex], [
    you, have, brass lantern], [you, have, glasses], [decoration, in, Cultural Complex], [
    you, have, cheaply-made sword], [you, have, Multi-Implementeers], [you, have, razor-
    like gloves], [glasses, is, clothing], [gloves, is, clothing], [sword, is, animate], [
    tunnel, is, animate], [sign, is, animate], [lantern, is, animate], [sword, is, equip],
     [lantern, is, equip]
```

Valid actions are defined by Hausknecht *et al.* (2020) as the set of actions guaranteed to cause a change in the current world state and are identified by the Jericho framework. For example in one particular state me might have the following valid actions:

```
Valid Actions: west, turn lantern off, east, south, put multi down, put forever down, put
    lantern down, put rune down, put glasses down, put sword down, take razor off, put on
    glasses, examine glasses, lower razor, throw multi, throw lantern, put multi in
    glasses, north
```

Tasks. Given this dataset, I focus on two tasks within it as formally defined by JerichoWorld. A successful world model will be able to accomplish both of these tasks.

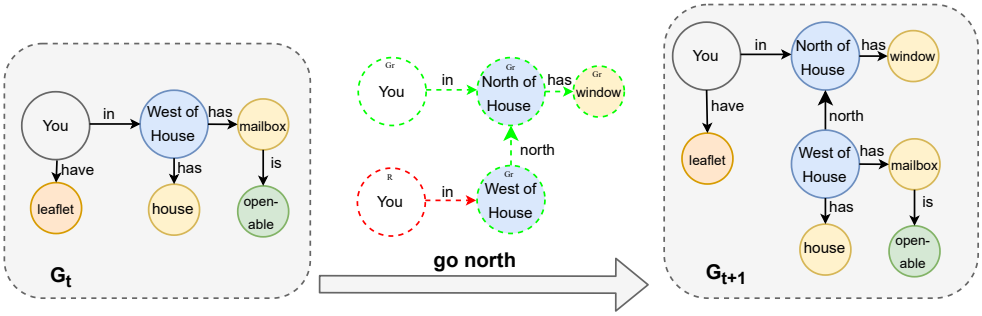


Figure 3.2: The transformation between subsequent world knowledge graphs G_t and G_{t+1} based on the states in Figure 3.1. The green (Gr) outlined portions in the center are additions to G_t to get G_{t+1} (i.e. $G_{t+1} - G_t$) and the red (R) portions similarly represent deletions to G_t (i.e. $G_t - G_{t+1}$).

1. **Knowledge Graph Generation:** this task involves predicting the graph at time step $t+1 : G_{t+1} \in S_{t+1}$ given the textual observations, valid actions, and graph at time step $t : O_t, V_t, G_t \in S_t$, and action A for all samples in the dataset.

2. **Valid Action Generation:** this task is formally defined as predicting the set of sequences of valid actions at time step $t+1 : V_{t+1} \in S_{t+1}$ given the textual observations, valid actions, and graph at time step $t : O_t, V_t, G_t \in S_t$, and action A for all samples in the dataset.

3.2 The Worldformer

This section describes the core methodological contributions of my work in creating world models for text games. I first show how knowledge graph generation can be simplified to predicting the graph difference between agent steps. I then describe the Worldformer, a transformer-based architecture, and end-to-end training method—including an objective function—that treats both of the world modeling tasks as a Set of Sequences generation problem.

3.2.1 Knowledge Graph Difference Generation

Figure 3.2 describes the gist of my simplification of the knowledge graph generation problem. Recall that knowledge graphs are directed graphs that are stored the form of a *set of tuples* as $\langle s, r, o \rangle$ such that s is a subject, r is a relation, and o is an object. Let the knowledge graphs representing the world state at two subsequent steps be G_t and G_{t+1}. At every step, tuples are either added or deleted from the graph G_t to update the belief state about the world and turn it into graph G_{t+1}. Using this observation, I can simplify the knowledge graph generation problem. Instead of predicting G_{t+1} given G_t and prior context, I can instead predict the *differences* between the two graphs.

In Figure 3.2, between steps t and $t+1$, I see that $G_{t+1} - G_t$ is the set of tuples that are added to G_t and $G_t - G_{t+1}$ the set of tuples are are deleted from G_t. Together they make up the graph differences. Here, I make a second key observation that allows for yet further simplification of the problem. This observation is based on generally applicable properties of such worlds: (1) locations are fixed and unique, i.e. the positions of locations with respect to each other does not change; (2) objects and characters can only be in one location at a time; and (3) contradicting object attributes can be identified using a lexical dictionary such as WordNet (Miller 1995), e.g. an object cannot be both open and closed at the same time. These properties let us uniquely identify the triples to be deleted from the graph $G_t - G_{t+1}$ given triples to be added to the graph $G_{t+1} - G_t$. Additional implementation details can be found in Appendix A.2.

Taken together, the Knowledge Graph Generation task can be cast as follows: predict the nodes to be added to the graph G_t at time step $t : G_{t+1} - G_t$ (a much smaller set than G_{t+1} by itself) to transform it into graph G_{t+1} given the textual observations, valid actions, and graph at time step $t : O_t, V_t, G_t \in S_t$, and action A for all samples in the dataset.

3.2.2 Multi-task Architecture

The Worldformer is a multi-task world model that simultaneously learns to perform both knowledge graph and valid action generation. It is built on the hypothesis that each of these tasks contains information crucial to the other—the valid actions that can be executed at any timestep are entirely dependent on the current state and vice versa the state knowledge graph updates on the basis of the previously executed action.

Figure 3.3 describes the architecture of the Worldformer. The inputs to the architecture are textual observations, valid actions, and graph at time step t : $O_t, V_t, G_t \in S_t$. O_t and V_t are encoded through a bidirectional text encoder into $\mathbf{O_t}$. In my work, I used an architecture similar to BERT (Devlin *et al.* 2019) with the original pre-trained weights that are then fine-tuned using a masked language model (MLM) loss on observations taken from the training data. $\mathbf{O_t}$ is the output of the final hidden layer. The graph encoder receives G_t and encodes it into $\mathbf{G_t}$. It is also similar to BERT, but is pre-trained on knowledge graphs found in the training data using a MLM loss with a phrase-level masking scheme where whole components of a $\langle s, r, o \rangle$ graph triple (individual underlined portions in Figure 3.3) are masked at once. Again, $\mathbf{G_t}$ is the output of the final hidden layer.

$\mathbf{O_t}$ and $\mathbf{G_t}$ are passed into a representation aggregator which then sends the combined encoded state representation $\mathbf{S_t}$ to one of two autoregressive decoders that have the same general internal architecture as GPT-2 (Radford *et al.* 2019). This aggregator is a small transformer sand-witched between linear layers that is meant to reduce the dimension of the input vector representation, more details are found in Appendix A. During training, the first decoder is conditioned on $\mathbf{S_t}$ directly and $\mathbf{O_t}$ through cross-attention and takes in the valid actions of the next state V_{t+1} as input, learning to predict the same input sequence shifted to the right as sequence-to-sequence models do. Similarly, the second decoder is conditioned on $\mathbf{S_t}$ directly and $\mathbf{G_t}$ through cross-attention and takes in the knowledge graph of the next state $G_{t+1} - G_t$ as input.

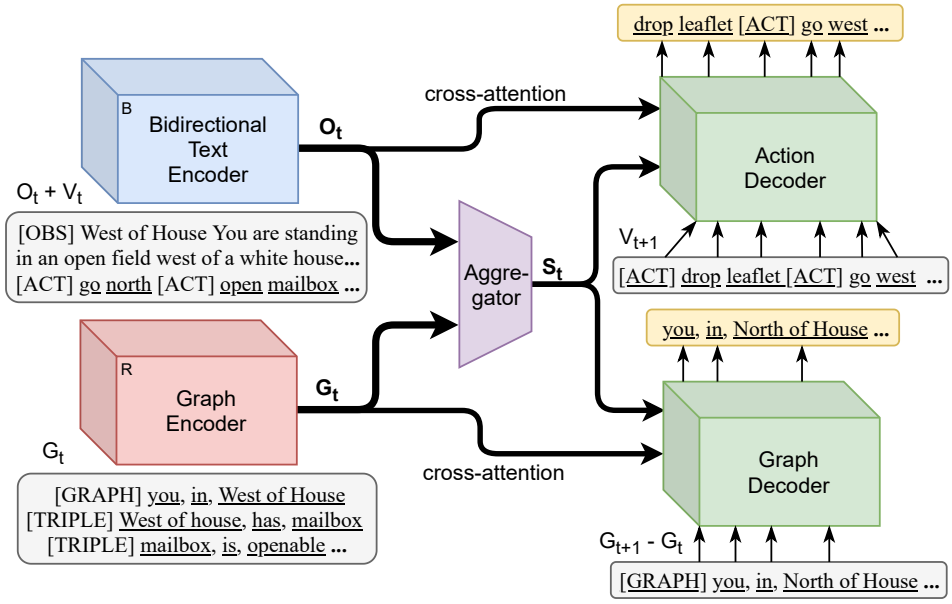


Figure 3.3: The Worldformer architecture. The text encoder (B) and graph encoder (R) have similar architecture but different pre-training strategies. Both the decoders are not pre-trained and have identical architectures. Solid black lines indicate gradient flow.

3.2.3 Set of Sequences Generation and Training

I observe that both the knowledge graph difference $G_{t+1} - G_t$ and the valid actions V_{t+1} are both *Sets of Sequences* where the ordering of the sequence of tokens within an action or a graph triple matters but the ordering of all the actions and triples does not. Standard autoregressive decoding used in sequence-to-sequence (Seq2Seq) models (Sutskever *et al.* 2014) does not account for such permutation invariance. I frame the graph and action prediction tasks as a generation of a Set of Sequences (SOS) problem—expanding on the simple set prediction problem definition proposed by works such as Deep Sets (Zaheer *et al.* 2017) to account for the specific structure of Sets of Sequences. This problem structure is used to then formulate a training methodology that lets autoregressive decoders better account for the SOS structure.

For both of the decoders in Figure 3.3, we are given a target sequence $Y = \{y_1, ..., y_M\}$ and some input context via the encoders X. Standard autoregressive techniques factor the distribution over the target sequence into a chain of conditional probabilities with a causal

left to right structure.

$$P(Y|X;\theta) = \prod_{i=1}^{M+1} p(y_i|y_{0:i-1}, X;\theta) \tag{3.1}$$

Where θ represents the overall network parameters. This can then be used to formulate a maximum likelihood training loss with cross-entropy at every step.

$$\mathcal{L}_{seq} = \log P(Y|X;\theta) = \sum_{i=1}^{M+1} \log p(y_i|y_{0:i-1}, X;\theta) \tag{3.2}$$

In my setting, we can group elements in Y into its set of sequences form

$$\begin{aligned} Y_{\text{sos}} = \{y'_1 ... y'_{M'}\}, y'_i \in V_{t+1} \text{ or } y'_i \in G_{t+1} - G_t, M' \leq M \\ \text{where } y'_i = \{y_k ... y_{k+l}\}, \sum_j \text{len}(y'_j) = M \end{aligned} \tag{3.3}$$

Via the decoders, we seek to learn a transformation from $\mathbf{S_t} \in \mathbb{R}^d$ (the input d-dimensional state representation vector) and $Y_{\text{sos}} \in \mathcal{Y}$ (decoder inputs in the space of all possible decoder inputs $\mathcal{Y}$) that map to the permutation invariant target set of sequences Y_{sos}. This function can then be defined as $f : \mathbb{R}^d \cup 2^{\mathcal{Y}} \rightarrow 2^{\mathcal{Y}}$ as the permutation invariance of part of the domain and range of this function makes it the power set of $\mathcal{Y}$.

Combining this definition of permutation invariant functions with Eq. 3.2, 3.3, we can factorize the distribution over the output Set of Sequences as the following chain of probabilities:

$$P(Y_{\text{sos}}|X;\theta) = \prod_{i=1}^{M+1} p(y'_i|X;\theta) \tag{3.4a}$$

$$p(y'_i|X;\theta) = \prod_{k=l}^{l+n} p(y_k|y_{l:k-1}, X;\theta) \tag{3.4b}$$

$$\text{where } l = \sum_{j<i} \text{len}(y'_j),\ n = \text{len}(y'_i)$$

With the key intuition here being that Eq. 3.4a factorizes the distribution such that each

element of Y_{sos} is independent of other elements in the set, but tokens of an element y'_i in the set are conditioned on preceding tokens within the element (Eq. 3.4b).

This in turn gives us a maximum likelihood Set of Sequences loss that can be used to train a model to output a Set of Sequences.

$$\mathcal{L}_{\text{sos}} = \log P(Y_{\text{sos}}|X;\theta)$$

$$\mathcal{L}_{sos} = \sum_{i=1}^{M+1} \log p(y'_i|X;\theta) \tag{3.5a}$$

$$\mathcal{L}_{sos} = \sum_{i=1}^{M+1} \sum_{k=l}^{l+n} \log p(y_k|y_{l:k-1}, X;\theta)$$

$$\text{where } l = \sum_{j<i} \text{len}(y'_j),\ n = \text{len}(y'_i) \tag{3.5b}$$

In my formulation, I have observation sequences at timestep t : O_t, V_t encoded into $\mathbf{O_t}$, graph G_t encoded into $\mathbf{G_t}$, and all of them combined into $\mathbf{S_t}$, with the output Sets of Sequences at timestep $t+1$ being the graph difference $G_{t+1} - G_t$ and valid actions V_{t+1}. Across the two decoders, this gives us a combined loss:

$$\mathcal{L}_{\text{world}} = \log P(G_{t+1} - G_t|\mathbf{S_t}, \mathbf{G_t};\theta) + \log P(V_{t+1}|\mathbf{S_t}, \mathbf{O_t};\theta) \tag{3.6}$$

This loss is used to multi-task train the Worldformer simultaneously across the two tasks.

3.3 Evaluation

I evaluate the Worldformer by comparing it on both of the tasks across 9 never-before-seen testing games against strong baselines. I further present ablation studies in each task to determine the relative importance of each of the techniques presented in the previous section.

3.3.1 Knowledge Graph Generation

All sequence models use a fixed graph vocabulary of size 7002 that contains all unique relations and entities at train and test times.

Metrics. For this task, I report two types of metrics (Exact Match or EM and F1) operating on two different levels—at a *graph* tuple level and another at a *token* level. EM checks for direct overlap between the predictions and ground truth, while F1 is a harmonic mean of predicted precision and recall. The graph level metrics are based on matching the set of $\langle$*subject, relation, object*$\rangle$ triples within the graph, all three tokens in a particular triple must match a triple within the ground truth graph to count as a true positive. The token level metrics operate on measuring unigram overlap in the graphs, any relations or entities in the predicted tokens that match the ground truth count towards a true positive.

Baselines

I compare the Worldformer to 4 baselines taken from contemporary knowledge graph-based world modeling approaches in text games—three of which have been developed and introduced by me.

Rules. Following Ammanabrolu and Hausknecht (2020) (or as seen in Chapter 4), I extract graph information from the observation using information extraction tools such as OpenIE (Angeli *et al.* 2015) in addition to some hand-authored rules to account for the irregularities of text games.

At every step, given the current state and possible attributes as context. The rest of the triples are extracted using OpenIE (Angeli *et al.* 2015).

- Linking the current room type (e.g. "Kitchen", "Cellar") to the items found in the room with the relation "has", e.g. $\langle kitchen, has, lamp \rangle$
- All attribute information for each object is linked to the object with the relation "is". e.g. $\langle egg, is, treasure \rangle$

- Linking all inventory objects with relation "have" to the "you" node, e.g. $\langle you, have, sword \rangle$
- Linking rooms with directions based on the action taken to move between the rooms, e.g. $\langle Behind\ House, east\ of, Forest \rangle$ after the action "go east" is taken to go from behind the house to the forest

Question-Answering. This baseline comes from the Q*BERT agent described in Ammanabrolu *et al.* (2020d) (or as seen in Chapter 4). It is trained on the SQuAD 2.0 (Rajpurkar *et al.* 2018), the Jericho-QA text game question answering dataset (Ammanabrolu *et al.* 2020d) on the same set of training games as found in Worldformer, and then on Worldformer itself by formatting my dataset in the style of questions and answers when possible. It uses the ALBERT (Lan *et al.* 2020) variant of the BERT (Devlin *et al.* 2019) natural language transformer to answer questions and populate the knowledge graph via a few hand-authored rules from the answers. Examples of questions asked include: "What is my current location?", "What objects are around me?".

Seq2Seq. I introduce an encoder-decoder based sequence-to-sequence learning approach (Sutskever *et al.* 2014) inspired by the transformer model BART (Lewis *et al.* 2020). The model architecture consists of a bidirectional encoder such as BERT (Devlin *et al.* 2019) that takes the full set of textual observations—including location and inventory descriptions—as input and an autoregressive decoder such as GPT-2 (Radford *et al.* 2019) which takes in the current graph and learns to predict the graph sequence shifted by a token. The weights of the encoder are fine-tuned from BERT's original weights on both the graphs, in triple form, and the textual observations taken from the training games using a masked language modeling loss. The decoder is not pre-trained. During test time, only the starting token is given to the decoder and it decodes the graph token by token via bean search until an end-of-sequence token is reached.

GATA-World. I adapt the Graph-Aided Transformer Agent (Adhikari *et al.* 2020) to my task. It consists of the same encoder structure as the Worldformer but contains one decoder that performs single-task Seq2Seq learning to decode both the set of tuples that

Table 3.1: Results across both the tasks for the models specified. Overall indicates a size weighted average. All experiments were performed over three random seeds, with standard deviations not exceeding ± 3.2 in any of the overall categories for KG prediction and ± 1.2 for valid action prediction. Bolded results indicate highest overall scores. Asterisk ($*$) indicates when the top result is significantly higher ($p < 0.05$ with an ANOVA test followed by a post-hoc pair-wise Tukey test) over all alternatives. † indicates this result is *not* significantly higher than Worldformer.

Expt.	Metrics	Game	zork1	lib.	det.	bal.	pent.	ztuu	ludi.	deep.	temp.	overall
		Size	886	654	434	990	276	462	2210	630	1294	7836
Knowledge Graph Prediction												
Rules	**Gr.**	**EM**	3.72	7.61	1.39	9.17	6.44	4.94	5.10	0.49	2.48	4.70
		F1	4.46	12.87	4.55	11.90	10.22	10.06	8.37	0.64	3.36	7.25
	Tok.	**EM**	6.08	10.33	7.51	32.53	16.48	14.40	14.47	3.34	7.42	13.08
		F1	8.42	26.74	10.23	36.09	23.36	21.74	18.48	3.86	9.44	17.50
Q*BERT (Question Answering)	**Gr.**	**EM**	24.56	29.14	34.45	41.22	28.96	22.17	41.44	4.42	36.84	32.79
		F1	24.88	31.46	36.23	41.85	30.12	26.26	46.74	4.66	39.86	35.48
	Tok.	**EM**	43.93	49.78	60.28	85.81	65.02	49.44	57.58	9.31	48.98	**53.58**†
		F1	48.31	52.76	63.21	86.18	69.54	49.82	60.95	9.84	49.17	**55.74**†
Seq2Seq	**Gr.**	**EM**	12.44	18.42	26.86	8.19	22.18	16.89	12.94	8.38	16.48	14.29
		F1	12.96	18.89	29.48	9.04	23.54	16.89	14.18	10.47	18.52	15.54
	Tok.	**EM**	18.01	20.26	35.86	17.60	25.48	17.19	14.8	13.25	22.48	18.80
		F1	21.12	20.84	35.86	18.86	27.72	17.87	15.42	13.25	24.34	19.96
GATA-W	**Gr.**	**EM**	22.30	24.72	21.72	23.68	22.81	27.00	24.55	23.76	24.52	24.06
		F1	25.34	26.47	22.14	26.54	27.63	27.00	24.55	23.76	24.92	25.19
	Tok.	**EM**	33.09	33.88	25.64	34.64	37.71	35.81	35.94	32.48	40.89	35.31
		F1	33.93	34.86	25.80	38.68	39.59	38.88	37.16	32.48	43.97	37.10
Worldformer	**Gr.**	**EM**	21.62	34.39	41.05	50.41	30.00	41.56	40.10	41.87	42.43	**39.15***
		F1	24.44	34.39	44.53	52.43	34.30	42.20	41.65	42.74	45.17	**41.06***
	Tok.	**EM**	42.88	41.98	54.39	62.22	49.00	50.80	51.29	50.04	53.81	51.32
		F1	48.12	41.98	59.13	62.22	49.00	52.24	51.29	50.04	54.96	52.45
Valid Action Prediction												
Seq2Seq	**Act**	**EM**	16.65	15.13	18.19	16.19	23.39	14.75	20.10	14.71	20.34	18.10
		F1	17.85	16.88	21.12	18.23	25.87	15.13	20.86	14.86	22.14	19.44
CALM	**Act**	**EM**	18.67	11.18	17.37	10.04	13.77	11.29	15.49	10.31	13.13	13.79
		F1	18.90	25.49	34.42	12.16	34.40	9.95	20.94	7.84	18.57	19.11
World-former	**Act**	**EM**	23.08	22.55	20.97	29.08	27.05	20.71	21.36	24.04	22.80	**23.22***
		F1	23.50	26.52	25.28	32.89	31.32	23.66	22.27	26.12	25.66	**25.54***

must be added as well as deleted from the graph in the form of: $\langle$*add, node1, node2, relation*$\rangle$ or $\langle$*del, node1, node2, relation*$\rangle$. This is equivalent to predicting $(G_{t+1} - G_t) \cup (G_t - G_{t+1})$. It is trained with the Seq2Seq cross-entropy loss (Eq. 3.2).

Analysis Table 3.1 describes the results in this task over all the games. I see that on the graph level metrics, the Worldformer performs significantly better than all other other baselines. On the token level metrics, the Worldformer and QA method are comparable—the difference between these two methods are statistically non-significant ($p = 0.18$) with each other but both significantly ($p < 0.05$) higher than all others. The QA method, and other extractive methods, highlight portions of the input observation that form the graph and are particularly well suited for the token level metrics. The JerichoWorld developers note

Table 3.2: Worldformer ablations to test the impact of its three main components for KG prediction. All results are size weighted averages over all test games over three random seeds, with standard deviations not exceeding ± 3.2 in any category.

Ablation			**Graph**		**Token**	
Graph Diff	**Multi Task**	**SOS Loss**	**EM**	**F1**	**EM**	**F1**
			14.29	15.54	18.80	19.96
	✓	✓	29.29	31.41	39.99	41.02
✓		✓	32.60	34.65	42.74	44.35
✓	✓		35.94	36.17	48.82	50.18
✓	✓	✓	**39.15**	**41.06**	**51.32**	**54.45**

that these approaches are prone to over-extraction, i.e. extracting more text than is strictly relevant from the input observation aiding token level overlap but resulting in a sharp drop in terms of the graph level metrics. Recall that text games are *partially observable* and so the textual observations themselves may potentially be incomplete. An example of such an observation is: *"You see a locked chest in front of you in the cellar."*. The ground truth graph for this would be: ⟨*you, in, Cellar*⟩, ⟨*chest, in, cellar*⟩, ⟨*chest, is, lockable*⟩, ⟨*sword, in, chest*⟩. The last fact in the graph, the sword being in the chest, is not revealed to you via the observation until you open the chest and thus cannot be predicted by extractive approaches like Rules and QA. The Worldformer is able to make a informed guess as to the contents of the chest due to its training, providing a form of look ahead that the Rules and QA systems cannot.

Table 3.3 present the results of an ablation study testing the relative importance of the three main components of the Worldformer: graph difference prediction, multi-task training, and the SOS loss. I note that a model without any of these components is equivalent to the Seq2Seq approach described previously. I see significant drops in performance, particularly on the graph level metrics, when any single one of these components are removed. This indicates that all three components are necessary for the Worldformer to achieve state-of-the-art performance.

In particular, I note that the largest performance drop was when Worldformer did not

use the graph difference simplification. In this case, the KG prediction task is simplified to predicting only; the length of the set of sequences $G_{t+1} - G_t$ is much smaller than G_{t+1}. There are on average 3.42 triples or 10.42 tokens per state across the JerichoWorld test dataset for $G_{t+1} - G_t$ but a mean of 8.71 triples or 26.13 tokens per state for G_{t+1}. This also explains the increased performance of the GATA-W over the baseline Seq2Seq agent—this agent only needs to predict on average 5.04 rules or 20.16 tokens across the testing games. Predicting a smaller number of triples and tokens per state makes the problem relatively more tractable for world modeling agents. Qualitative examples illustrating these trends are found in Appendix A.3.1.

3.3.2 Valid Action Generation

Similarly to the other task, I compare the Worldformer to an existing baseline for valid action prediction. All models use a fixed vocabulary of size 11,056 at train and test times.

Metrics. For this task, I adapt the graph level Exact Match (EM) and F1 metrics as described in the previous task to actions. In other words, positive EM or F1 happens only when all tokens in a predicted valid action match one in the gold standard set. Given that most valid actions have less than four tokens, I do not use standard Seq2Seq metrics—such as BLEU (Papineni *et al.* 2002)—intended for measuring n-gram overlap in longer sequences.

Seq2Seq. This single-task model is identical to the Seq2Seq model described in the previous task but is single-task trained to predict valid actions.

CALM. I would like to note the presence of a complementary dataset of observation-action pairs created by humans on the ClubFloyd online Interactive Narrative forum.[2] This dataset appears in both Ammanabrolu and Hausknecht (2020) and Yao *et al.* (2020) with the latter using it to tune a GPT-2 model for valid action prediction using a GPT-2 based Seq2Seq valid action model dubbed CALM.[3] This model takes in O_t, A, O_{t+1} and attempts

[2] http://www.allthingsjacq.com/interactive_fiction.html
[3] https://github.com/princeton-nlp/calm-textgame

Table 3.3: Worldformer ablations to test the impact of its two main components for action prediction. All results are size weighted averages over all test games over three random seeds, with standard deviations not exceeding ± 1.2 in any category.

Ablation		**Act**	
Multi Task	**SOS Loss**	**EM**	**F1**
		18.10	19.44
	✓	20.78	22.42
✓		20.12	21.28
✓	✓	**23.22**	**25.54**

to output V_{t+1}.

In Table 3.1, we see that the Worldformer significantly outperforms the Seq2Seq baseline on all the games and CALM overall. Each valid action in a text game requires at most 5 tokens. This combined with an average of 10.30 valid actions per test state means that for every state I would need to generate about 52 tokens. Yet further, the vocabulary size for actions is $11,056$, larger than the graph vocabulary of $7,002$. This increase in task difficulty explains the relative decrease in the magnitude of performance metrics between KG and valid action prediction tasks.

Both the Seq2Seq model and CALM—which is trained on a different dataset—are comparable on F1 scores but Seq2Seq is better overall for exact matches. CALM also has relatively higher variance in performance across the test games than the other two methods—e.g. on some games such as *zork1* and *detective* it outperforms the Seq2Seq and is not too far off the Worldformer especially in terms of F1 score. This would appear to indicate that the Club Floyd dataset of text game transcripts that CALM was trained on is better suited for transfer to certain games than others, likely due to differences in training set genre similarities. A careful mix of these datasets could potentially lead to greater generalization performance, though this is left to future work.

Table 3.3 presents an ablation study that tests the two main components of the Worldformer for this task: multi-task learning, and SOS loss. As with the KG prediction task, I observe significant drops in performance when either of these components are taken

away—suggesting that they are relatively critical components.

There is a correlation between performance of the baseline Seq2Seq model to the average number of valid actions for the testing game (see Appendix A). This is likely due to label imbalance in the dataset, the model likely learns a common set of actions found across all games such as navigation actions before learning more fine-grained actions. E.g., in almost every single instance, the standard movement actions (north, south, east, west) are all predicted regardless of which directional actions are actually available—likely due to all of them being very commonly available together on average across the training data. Another example, *ztuu, deephome,* and *balances* have a high number of gold standard average valid actions while *pentari, ludicorp, detective,* and *temple* which have a low number of average valid actions. While the latter set of games have generally higher performance on both the Seq2Seq and Worldformer models, the gap is significantly less pronounced with the Worldformer. I hypothesize that this is due to the multi-task training of the Worldformer—encoder representations now contain enough information regarding the next knowledge graph to alleviates the label imbalance of the actions and enable prediction of more fine-grained actions. Again, qualitative samples illustrating these trends may be found in Appendix A.3.1.

3.4 Conclusions

This chapter presents the Worldformer dataset and corresponding benchmarks that seek to drive progress in textual world modeling. This primarily involves two key challenges behind the creation of agents that can understand and generate natural language in a diverse set of interactive and situated settings such as text games. My dataset provides mappings from textual observations to ground truth knowledge graph states to enable agents to learn to infer the state of the world—alleviating the *knowledge representation* or *Textual-SLAM* challenge. A key insight from an comparison of baseline models shows that a promising future direction lies in *inferring* the knowledge graph world state through commonsense

reasoning rather than *extracting* this information due to the partial observability of text games.

A second world modeling task revolves around tacking the *combinatorially-sized action space* of text games. The dataset also provides mappings from textual observations to valid actions—i.e. the set of contextually relevant actions guaranteed to change the world in any state. A qualitative analysis of a state-of-the-art Seq2Seq model adapted to the domain and trained for this task suggests that while learning to conditionally generate commonly occurring actions across a large set of games might be relatively easy, learning to generate specific and contextually relevant actions provides a significantly more difficult challenge.

In particular, the Worldformer's state-of-the-art performance and the ablation studies have three potential implications: (1) the simplification of the knowledge representation problem into that of predicting knowledge graph differences between subsequent states is a critical step in making the problem more tractable; (2) performance improvements due to multi-task training imply that acting in and mapping these worlds is inherent a highly correlated problem and benefits from being solved jointly; and (3) the performance boosts due to the SOS loss suggest that accounting for this property of graphs and actions enables more effective training than if we were to treat them as simple sequences.

CHAPTER 4
SCALING TO COMBINATORIAL LANGUAGE ACTION SPACES

To gain a sense for the challenges surrounding natural language generation, we need to first understand how large this space really is. In order to solve solve a popular IF game such as *Zork1* it's necessary to generate actions consisting of up to five-words from a relatively modest vocabulary of 697 words recognized by Zork's parser. Even this modestly sized vocabulary leads to $\mathcal{O}(697^5) = 1.64 \times 10^{14}$ possible actions at every step—a dauntingly-large *combinatorially-sized action space* for a learning agent to explore.

This chapter introduces three novel agents that utilizes both a knowledge graph based state space and shows how to train such agents—using both off and on policy reinforcement learning. I then conduct an empirical study evaluating my agent across a diverse set of IF games followed by an ablation analysis studying the effectiveness of various components of my algorithm as well as its overall generalizability. Remarkably I show that the agents achieve state-of-the-art performance on a large proportion of the games despite the exponential increase in action space size.

4.1 Off Policy: Knowledge Graph Deep Q-Network

This section introduces an off-policy RL algorithm that can train agents to play text games, the KG-DQN. I use all the same POMDP definitions as previously introduced. Following Narasimhan *et al.* (2015), all actions A that will be accepted by the game's parser are available to the agent at all times. When playing the game, the agent chooses an action and receives an observation o_t from the simulator, which is a textual description of the current game state. The state graph G_t is updated according to the given observation, as seen in Figure 4.1 and Chapter 8.

I use the Q-Learning technique (Watkins and Dayan 1992) to learn a control policy

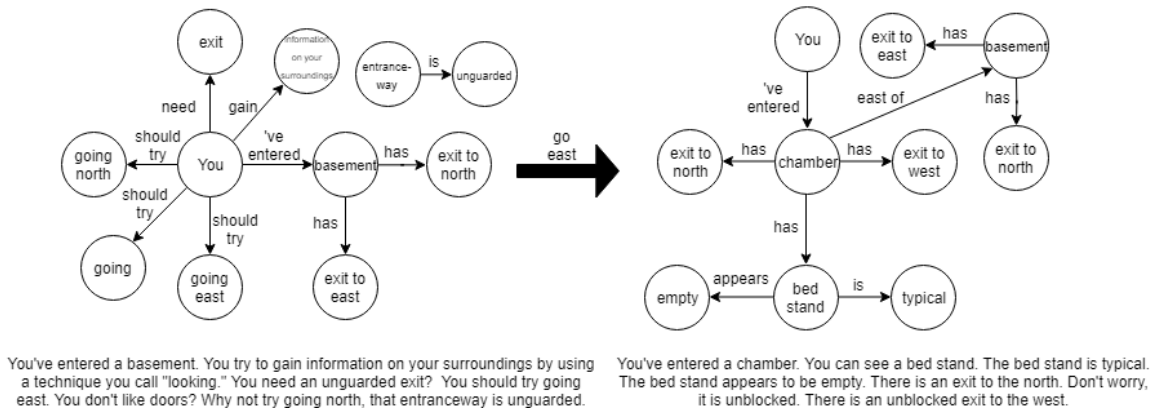


Figure 4.1: Graph update rules

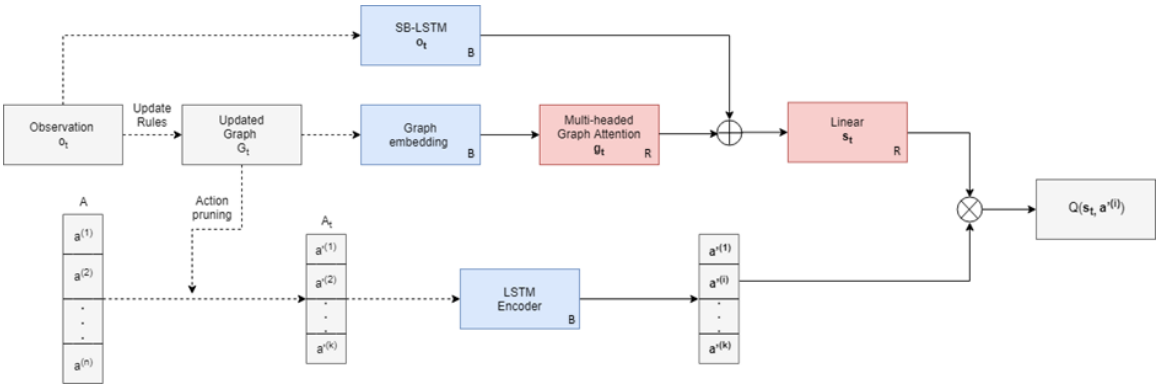


Figure 4.2: KG-DQN architecture, blue shading (or the symbol 'B') indicates components that can be pre-trained and red (or the symbol 'R') indicates no pre-training. The solid lines indicate gradient flow for learnable components.

$\pi(a_t|s_t)$, $a_t \in A$, which gives us the probability of taking action a_t given the current state s_t. The policy is determined by the Q-value of a particular state-action pair, which is updated using the Bellman equation (Sutton and Barto 2018):

$$Q_{t+1}(s_{t+1},a_{t+1}) = E[r_{t+1} + \gamma \max_{a \in A_t} Q_t(s,a)|s_t,a_t] \tag{4.1}$$

where γ refers to the discount factor and r_{t+1} is the observed reward. The policy is thus to take the action that maximizes the Q-value in a particular state, which will correspond to the action that maximizes the reward expectation given that the agent has taken action a_t at the current state s_t and folloId the policy $\pi(a|s)$ after.

The architecture in Figure 4.2 is responsible for computing the representations for both the state s_t and the actions $a^{(i)} \in A$ and coming to an estimation of the Q-value for a particular state and action. During the forward activation, the agent uses the observation to update the graph G_t using the rules outlined in Section 4.1.1.

The graph is then embedded into a single vector $\mathbf{g_t}$. I use Graph Attention (Veličković *et al.* 2018) with an attention mechanism similar to that described in Bahdanau *et al.* (2014). Formally, the Multi-headed Graph Attention component receives a set of node features $H = \{\mathbf{h_1}, \mathbf{h_2}, \ldots, \mathbf{h_N}\}$, $\mathbf{h_i} \in \mathbb{R}^{\mathrm{F}}$, where N is the number of nodes and F the number of features in each node, and the adjacency matrix of G_t. Each of the node features consist of the averaged word embeddings for the tokens in that node, as determined by the preceding graph embedding layer. The attention mechanism is set up using self-attention on the nodes after a learnable linear transformation $W \in \mathbb{R}^{2\mathrm{F}\times\mathrm{F}}$ applied to all the node features:

$$e_{ij} = LeakyReLU(\mathbf{p} \cdot W(\mathbf{h_i} \oplus \mathbf{h_j})) \tag{4.2}$$

where $\mathbf{p} \in \mathbb{R}^{2\mathrm{F}}$ is a learnable parameter. The attention coefficients α_{ij} are then computed by normalizing over the choices of $k \in \mathcal{N}$ using the softmax function. Here $\mathcal{N}$ refers to the neighborhood in which I compute the attention coefficients. This is determined by the adjacency matrix for G_t and consists of all third-order neighbors of a particular node.

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k\in\mathcal{N}} exp(e_{ik})} \tag{4.3}$$

Multi-head attention is then used, calculating multiple independent attention coefficients. The resulting features are then concatenated and passed into a linear layer to determine $\mathbf{g_t}$:

$$\mathbf{g_t} = f(W_g(\|_{k=1}^{K}\sigma(\sum_{j\in\mathcal{N}} \alpha_{ij}^{(k)}\mathbf{W}^{(k)}\mathbf{h}_j)) + b_g) \tag{4.4}$$

where k refers to the parameters of the k^{th} independent attention mechanism, W_g and b_g the

weights and biases of this component's output linear layer, and $\|$ represents concatenation.

Simultaneously, an encoded representation of the observation $\mathbf{o_t}$ is computed using a Sliding Bidirectional LSTM (SB-LSTM). The final state representation $\mathbf{s_t}$ is computed as:

$$\mathbf{s_t} = f(W_l(\mathbf{g_t} \oplus \mathbf{o_t}) + b_l) \tag{4.5}$$

where W_l, b_l represent the final linear layer's weights and biases and $\mathbf{o_t}$ is the result of encoding the observation with the SB-LSTM.

The entire set of possible actions A is pruned by scoring each $a \in A$ according to the mechanism previously described using the newly updated G_{t+1}. I then embed and encode all of these action strings using an LSTM encoder (Sutskever *et al.* 2014). The dashed lines in Figure 4.2 denotes non-differentiable processes.

The final Q-value for a state-action pair is:

$$Q(\mathbf{s_t}, \mathbf{a_t}) = \mathbf{s_t} \cdot \mathbf{a_t} \tag{4.6}$$

This method of separately computing the representations for the state and action is similar to the approach taken in the DRRN (He *et al.* 2016b).

I train the network using experience replay (Lin 1993) with prioritized sampling (cf., Moore and Atkeson (1993)) and a modified version of the ϵ-greedy algorithm (Sutton and Barto 2018) that I call the ϵ_1, ϵ_2-greedy learning algorithm. The experience replay strategy finds paths in the game, which are then stored as transition tuples in a experience replay buffer D. The ϵ_1, ϵ_2-greedy algorithm explores by choosing actions randomly from A with probability ϵ_1 and from A_t with a probability ϵ_2. The second threshold is needed to account for situations where an action must be chosen to advance the quest for which the agent has no prior in G_t. That is, action pruning may remove actions essential to quest completion because those actions involve combinations of entities that have not been encountered before.

I then sample a mini-batch of transition tuples consisting of $\langle \mathbf{s_k}, \mathbf{a_k}, r_{k+1}, \mathbf{s_{k+1}}, \mathbf{A_{k+1}}, p_k \rangle$ from D and compute the temporal difference loss as:

$$L(\theta) = r_{k+1} + \gamma \max_{\mathbf{a} \in \mathbf{A_{k+1}}} Q(\mathbf{s_t}, \mathbf{a}; \theta) - Q(\mathbf{s_t}, \mathbf{a_t}; \theta) \tag{4.7}$$

Replay sampling from D is done by sampling a fraction ρ from transition tuples with a positive reward and $1 - \rho$ from the rest. As shown in Narasimhan *et al.* (2015), prioritized sampling from experiences with a positive reward helps the deep Q-network more easily find the sparse set of transitions that advance the game. The exact training mechanism is described in Algorithm 1.

4.1.1 Action Pruning

The number of actions available to an agent in a text adventure game can be quite large: $A = \mathcal{O}(|V| \times |O|^2)$ where V is the number of action verbs, and O is the number of distinct objects in the world that the agent can interact with, assuming that verbs can take two arguments. Some actions, such as movement, inspecting inventory, or observing the room, do not have arguments.

The knowledge graph is used to prune the combinatorially large space of possible actions available to the agent as follows. Given the current state graph representation G_t, the action space is pruned by ranking the full set of actions and selecting the top-k. My action scoring function is:

- +1 for each object in the action that is present in the graph; and
- +1 if there exists a valid directed path between the two objects in the graph.

I assume that each action has at most two objects (for example inserting a key in a lock).

Algorithm 2 ϵ_1, ϵ_2-greedy learning algorithm for KG-DQN

for episode=1 to M **do**
  Initialize action dictionary A and graph G_0
  Reset the game simulator
  Read initial observation o_1
  $G_1 \leftarrow updateGraph(G_0, o_1)$; $A_1 \leftarrow pruneActions(A, G_0)$ ▷ Section 4.1.1
  for step t=1 to T **do**
    if $random() < \epsilon_1$ **then**
      if $random() < \epsilon_2$ **then**
        Select random action $a_t \in A$
      else
        Select random action $a_t \in A_t$
    else
      Compute $Q(\mathbf{s_t}, \mathbf{a^{(i)}}; \theta)$ for $a^{(i)} \in A$ for network parameters θ ▷ Section 4.1, Eq. 4.6
      Select a_t based on $\pi(a|s_t)$
    Execute action a_t in the simulator and observe reward r_t
    Receive next observation $ot + 1$
    $G_{t+1} \leftarrow updateGraph(G_t, o_{t+1})$; $A_{t+1} \leftarrow pruneActions(A, G_{t+1})$ ▷ Figure 4.1
    Compute $\mathbf{s_{t+1}}$ and $\mathbf{A_{t+1}} = \{\mathbf{a'^{(i)}}$ for all $a'^{(i)} \in A\}$ ▷ Section 4.1
    Set priority $p_t = 1$ if $r_t > 0$, else $p_t = 0$
    Store transition $(\mathbf{s_t}, \mathbf{a_t}, r_t, \mathbf{s_{t+1}}, \mathbf{A_{t+1}}, p_t)$ in replay buffer D
    Sample mini-batch of transitions $(\mathbf{s_k}, \mathbf{a_k}, r_k, \mathbf{s_{k+1}}, \mathbf{A_{k+1}}, p_k)$ from D, with fraction ρ having $p_k = 1$
    Set $y_k = r_k + \gamma \max_{\mathbf{a} \in \mathbf{A_{k+1}}} Q(\mathbf{s_t}, \mathbf{a}; \theta)$, or $y_k = r_k$ if s_{k+1} is terminal
    Perform gradient descent step on loss function $L(\theta) = (y_k - Q(\mathbf{s_t}, \mathbf{a_t}; \theta))^2$

4.1.2 Benefits of a Persistent Memory

I conducted experiments in the TextWorld framework (Côté *et al.* 2018) using their "home" theme. TextWorld uses a grammar to randomly generate game worlds and quests with given parameters. Games generated with TextWorld start with a zero-th observation that gives instructions for the quest; I do not allow my agent to access this information. The TextWorld API also provides a list of *admissible* actions at each state—the actions that can

Table 4.1: Generated game details.

	Small	**Large**
Rooms	10	20
Total objects	20	40
Quest length	5	10
Branching factor	143	562
Vocab size	746	819
Average words per obs.	67.5	94.0
Average new RDF triples per obs.	7.2	10.5

be performed based on the objects that are present. I do not allow my agent to access the admissible actions.

I generated two sets of games with different random seeds, representing different game difficulties, which I denote as *small* and *large*. Small games have ten rooms and quests of length five and large games have tInty rooms and quests of length ten. Statistics on the games are given in Table 4.1. Quest length refers to the number of actions that the agent is required to perform in order to finish the quest; more actions are typically necessary to move around the environment and find the objects that need to be interacted with. The branching factor is the size of the action set A for that particular game.

The reward function provided by TextWorld is as follows: +1 for each action taken that moves the agent closer to finishing the quest; -1 for each action taken that extends the minimum number of steps needed to finish the quest from the current stage; 0 for all other situations. The maximum achievable reward for the small and large sets of games are 5 and 10 respectively. This allows for a large amount of variance in quest quality—as measured by steps to complete the quest—that receives maximum reward.

The following procedure for pre-training was done separately for each set of games. Pre-training of the SB-LSTM within the question-answering architecture is conducted by generating 200 games from the same TextWorld theme. The QA system was then trained on data from walkthroughs of a randomly-chosen subset of 160 of these generated games, tuned on a dev set of 20 games, and evaluated on the held-out set of 20 games. Table 4.2 provides details on the Exact Match (EM), precision, recall, and F1 scores of the QA system after training for the small and large sets of games. Precision, recall, and F1 scores are calculated by counting the number of tokens between the predicted answer and ground truth. An Exact Match is when the entire predicted answer matches with the ground truth. This score is used to tune the model based on the dev set of games.

A random game was chosen from the test-set of games and used as the environment for the agent to train its deep Q-network on. Thus, at no time did the QA system see the final

Table 4.2: Pre-training accuracy.

	EM	Precision	Recall	F1
Small	46.20	56.57	63.38	57.94
Large	34.13	52.53	64.72	55.06

testing game prior to the training of the KG-DQN network.

I compare my technique to three baselines:

Random command. This baseline samples from the list of admissible actions returned by the TextWorld simulator at each step.

LSTM-DQN. This was developed by Narasimhan *et al.* (2015).

Bag-of-Words DQN. This baseline uses a bag-of-words encoding with a multi-layer feed forward network instead of an LSTM.

To achieve the most competitive baselines, I used a randomized grid search to choose the best hyperparameters (e.g., hidden state size, γ, ρ, final ϵ, update frequency, learning rate, replay buffer size) for the BOW-DQN and LSTM-DQN baselines.

I tested three versions of my KG-DQN:

1. Un-pruned actions with pre-training
2. Pruned actions without pre-training
3. Pruned actions with pre-training (full)

My models use 50-dimensional word embeddings, 2 heads on the graph attention layers, mini-batch size of 16, and perform a gradient descent update every 5 steps taken by the agent.

All models are evaluated by observing the (a) time to reward convergence, and (b) the average number of steps required for the agent to finish the game with $\epsilon = 0.1$ over 5 episodes after training has completed. Following Narasimhan *et al.* (2015) I set ϵ to a non-zero value because text adventure games, by nature, require exploration to complete the quests. All results are reported based on multiple independent trials. For the large set

of games, I only perform experiments on the best performing models found in the small set of games. Also note that for experiments on large games, I do not display the entire learning curve for the LSTM-DQN baseline, as it converges significantly more slowly than KG-DQN. I ran each experiment 5 times and average the results.

Additionally, human performance on the both the games was measured by counting the number of steps taken to finish the game, with and without instructions on the exact quest. I modified Textworld to give the human players reward feedback in the form of a score, the reward function itself is identical to that received by the deep reinforcement learning agents. In one variation of this experiment, the human was given instructions on the potential sequence of steps that are required to finish the game in addition to the reward in the form of a score and in the other variation, the human received no instructions.

Recall that the number of steps required to finish the game for the oracle agent is 5 and 10 for the small and large maps respectively. It is impossible to achieve this ideal performance due to the structure of the quest. The player needs to interact with objects and explore the environment in order to figure out the exact sequence of actions required to finish the quest. To help benchmark my agent's performance, I observed people unaffiliated with the research playing through the same TextWorld "home" quests as the other models. Those who did not receive instructions on how to finish the quest never finished a single quest and gave up after an average of 184 steps on the small map and an average of 190 steps on the large map. When given instructions, human players completed the quest on the large map in an average of 23 steps, finishing the game with the maximum reward possible. Also note that none of the deep reinforcement learning agents received instructions.

On both small and large maps, all versions of KG-DQN tested converge faster than baselines (see Figure 4.3 for the small game and Figure 4.4 for the large game). I don't show BOW-DQN because it is strictly inferior to LSTM-DQN in all situations). KG-DQN converges 40% faster than baseline on the small game; both KG-DQN and the LSTM-DQN baseline reaches the maximum reward of five. On the large game, no agents achieve the

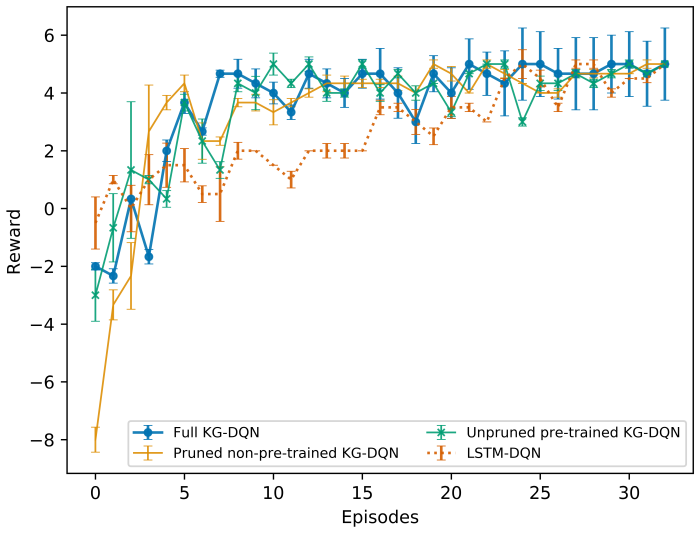


Figure 4.3: Reward learning curve for select experiments with the small games. Best vieId in color.

Table 4.3: Average number of steps (and standard deviation) taken to complete the small game.

Model	**Steps**
Random Command	319.8
BOW-DQN	83.1 ± 8.0
LSTM-DQN	72.4 ± 4.6
Unpruned, pre-trained KG-DQN	131.7 ± 7.7
Pruned, non-pre-trained KG-DQN	97.3 ± 9.0
Full KG-DQN	73.7 ± 8.5

maximum reward of 10, and the LSTM-DQN requires more than 300 episodes to converge at the same level as KG-DQN. Since all versions of KG-DQN converge at approximately the same rate, I conclude that the knowledge graph—i.e., persistent memory—is the main factor helping convergence time since it is the common element across all experiments.

After training is complete, I measure the number of steps each agent needs to complete each quest. Full KG-DQN requires an equivalent number of steps in the small game (Table 4.3) and in the large game (Table 4.4). Differences between LSTM-DQN and full KG-DQN are not statistically significant, $p = 0.199$ on an independent T-test. The ablated versions of KG-DQN—unpruned KG-DQN and non-pre-trained KG-DQN—require many

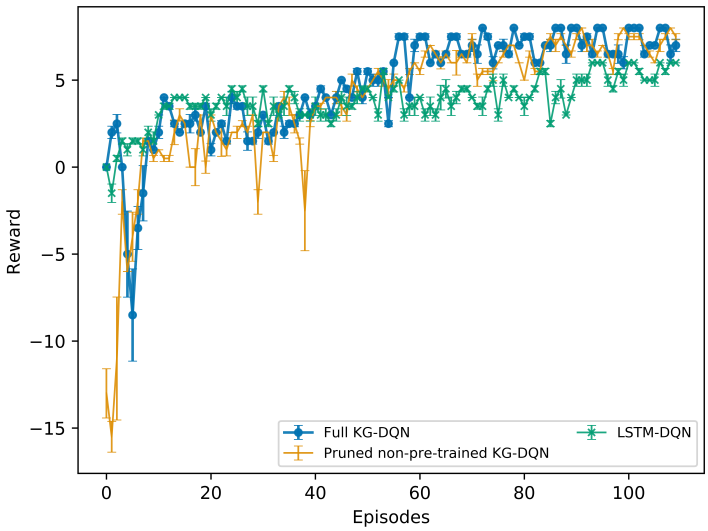


Figure 4.4: Reward learning curve for select experiments with the large games. Best vieId in color.

Table 4.4: Average number of steps (and standard deviation) taken to complete the large game.

Model	**Steps**
Random Command	2054.8
LSTM-DQN	260.3 ± 4.5
Pruned, non-pre-trained KG-DQN	340 ± 6.4
Full KG-DQN	265.9 ± 9.4

more steps to complete quests. TextWorld's reward function allows for a lot of exploration of the environment without penalty so it is possible for a model that has converged on reward to complete quests in as few as five steps or in many hundreds of steps. From these results, I conclude that the pre-training using my question-answering paradigm is allowing the agent to find a general understanding of how to pick good actions even when the agent has never seen the final test game. LSTM-DQN also learns how to choose actions efficiently, but this knowledge is captured in the LSTM's cell state, whereas in KG-DQN this knowledge is made explicit in the knowledge graph and retrieved effectively by graph attention. Taken together, KG-DQN converges faster without loss of quest solution quality.

I have shown that incorporating knowledge graphs into an deep Q-network can reduce

training time for agents playing text-adventure games of various lengths. I speculate that this is because the knowledge graph provides a persistent memory of the world as it is being explored. While the knowledge graph allows the agent to reach optimal reward more quickly, it doesn't ensure a high quality solution to quests. Action pruning using the knowledge graph and pre-training of the embeddings used in the deep Q-network result in shorter action sequences needed to complete quests.

The insight into pre-training portions of the agent's architecture is based on converting text-adventure game playing into a question-answering activity. That is, at every step, the agent is asking—and trying to answer—what is the most important thing to try. The pre-training acts as a form of transfer learning from different, but related games. However, question-answering alone cannot solve the text-adventure playing problem because there will always be some trial and error required.

4.2 On Policy: Knowledge Graph Advantage Actor Critic

Combining the knowledge-graph state space with a template action space, Knowledge Graph Advantage Actor Critic or KG-A2C, is an on-policy reinforcement learning agent that collects experience from many parallel environments. I first discuss the architecture of KG-A2C, then detail the training algorithm. As seen in Fig. 4.5, KG-A2C's architecture can broadly be described in terms of encoding a state representation and then using this encoded representation to decode an action. I describe each of these processes below.

Template Action Space. In order to reduce the size of this space while maintaining expressiveness, Jericho proposes the use of template-actions in which the agent first selects a template (e.g. $[put]$ __$[in]$__) then fills in the blanks using vocabulary words. There are 237 templates in *Zork1*, each with up to two blanks, yielding a template-action space of size $\mathcal{O}(237 \times 697^2) = 1.15 \times 10^8$. This space is six orders of magnitude smaller than the word-based space, but still six orders of magnitude larger than the action spaces used by previous text-based agents (Zahavy *et al.* 2018; Narasimhan *et al.* 2015). I demonstrate

how these templates provide the structure required to further constrain my action space via my knowledge graph—and make the argument that the *combination* of these approaches allows us to generate meaningful natural language commands.

Templates are subroutines used by the game's parser to interpret the player's action. They consist of interchangeable verbs phrases (VP) optionally followed by prepositional phrases ($VP\ PP$), e.g. *([carry/hold/take] __)* and *([drop/throw/discard/put] __ [at/against/on/onto] __)*, where the verbs and prepositions within [.] are aliases. As shown in Figure 4.6, actions may be constructed from templates by filling in the template's blanks using words in the game's vocabulary. Templates and vocabulary words are programmatically accessible through the Jericho framework and are thus available for every IF game.

Input Representation. The input representation network is broadly divided into three parts: an observation encoder, a score encoder, and the knowledge graph. At every step an observation consisting of several components is received: $o_t = (o_{t_{desc}}, o_{t_{game}}, o_{t_{inv}}, a_{t-1})$ corresponding to the room description, game feedback, inventory, and previous action, and total score R_t. The room description $o_{t_{desc}}$ is a textual description of the agent's location, obtained by executing the command "look." The game feedback $o_{t_{game}}$ is the simulators response to the agent's previous action and consists of narrative and flavor text. The inventory $o_{t_{inv}}$ and previous action a_{t-1} components inform the agent about the contents of its inventory and the last action taken respectively.

The observation encoder processes each component of o_t using a separate GRU encoder. As I am not given the vocabulary that o_t is comprised of, I use subword tokenization—specifically using the unigram subword tokenization method described in Kudo (2018). This method predicts the most likely sequence of subword tokens for a given input using a unigram language model which, in my case, is trained on a dataset of human playthroughs of IF games[1] and contains a total vocabulary of size 8000. For each of the GRUs, I pass in the final hidden state of the GRU at step $t-1$ to initialize the hidden state at step t. I

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

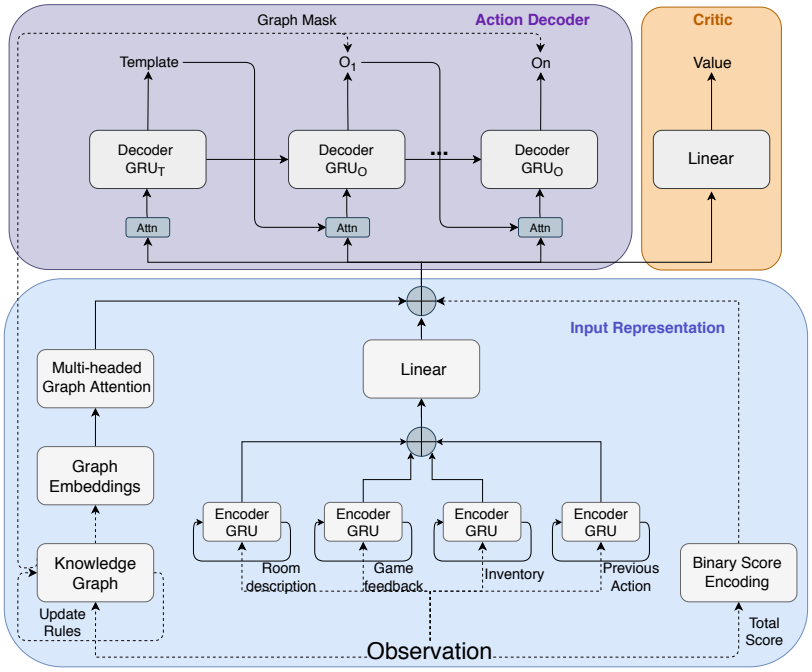


Figure 4.5: The full KG-A2C architecture. Solid lines represent computation flow along which the gradient can be back-propagated.

concatenate each of the encoded components and use a linear layer to combine them into the final encoded observation $\mathbf{o}_t$.

At each step, I update my knowledge graph G_t using o_t and it is then embedded into a single vector $\mathbf{g_t}$ as described in Chapter 3. I again use Graph Attention networks or GATs (Veličković *et al.* 2018) with an attention mechanism similar to that described in Bahdanau *et al.* (2014). Node features are computed as $H = \{\mathbf{h_1}, \mathbf{h_2}, \ldots, \mathbf{h_N}\}$, $\mathbf{h_i} \in \mathbb{R}^{\mathrm{F}}$, where N is the number of nodes and F the number of features in each node, consist of the average subword embeddings of the entity and of the relations for all incoming edges using my unigram language model. Self-attention is then used after a learnable linear transformation $W \in \mathbb{R}^{2\mathrm{F}\times\mathrm{F}}$ applied to all the node features. Attention coefficients α_{ij} are then computed by softmaxing $k \in \mathcal{N}$ with $\mathcal{N}$ being the neighborhood in which I compute

the attention coefficients and consists of all edges in G_t.

$$e_{ij} = LeakyReLU(\mathbf{p} \cdot W(\mathbf{h_i} \oplus \mathbf{h_j})) \tag{4.8}$$

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k \in \mathcal{N}} exp(e_{ik})} \tag{4.9}$$

where $\mathbf{p} \in \mathbb{R}^{2F}$ is a learnable parameter.

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k \in \mathcal{N}} exp(e_{ik})} \tag{4.10}$$

The final knowledge graph embedding vector $\mathbf{g_t}$ is computed as:

$$\mathbf{g_t} = f(W_g(\bigoplus_{k=1}^{K} \sigma(\sum_{j \in \mathcal{N}} \alpha_{ij}^{(k)} \mathbf{W}^{(k)} \mathbf{h}_j)) + b_g) \tag{4.11}$$

where k refers to the parameters of the k^{th} independent attention mechanism, W_g and b_g the weights and biases of the output linear layer, and $\bigoplus$ represents concatenation. The final component of state embedding vector is a binary encoding $\mathbf{c}_t$ of the total score obtained so far in the game—giving the agent a sense for how far it has progressed in the game even when it is not collecting reward. The state embedding vector is then calculated as $\mathbf{s_t} = \mathbf{g_t} \oplus \mathbf{o_t} \oplus \mathbf{c_t}$.

Action Decoder. The state embedding vector $\mathbf{s_t}$ is then used to sequentially construct an action by first predicting a template and then picking the objects to fill into the template using a series of Decoder GRUs. This gives rise to a template policy $\pi_{\mathbb{T}}$ and a policy for each object $\pi_{\mathbb{O}_i}$. Architecture wise, at every decoding step all previously predicted parts of the action are encoded and passed along with $\mathbf{s}_t$ through an attention layer which learns to attend over these representations—conditioning every predicted object on all the previously predicted objects and template. All the object decoder GRUs share parameters while the template decoder GRU_T remains separate.

To effectively constrain the space of template-actions, I introduce the concept of a *graph*

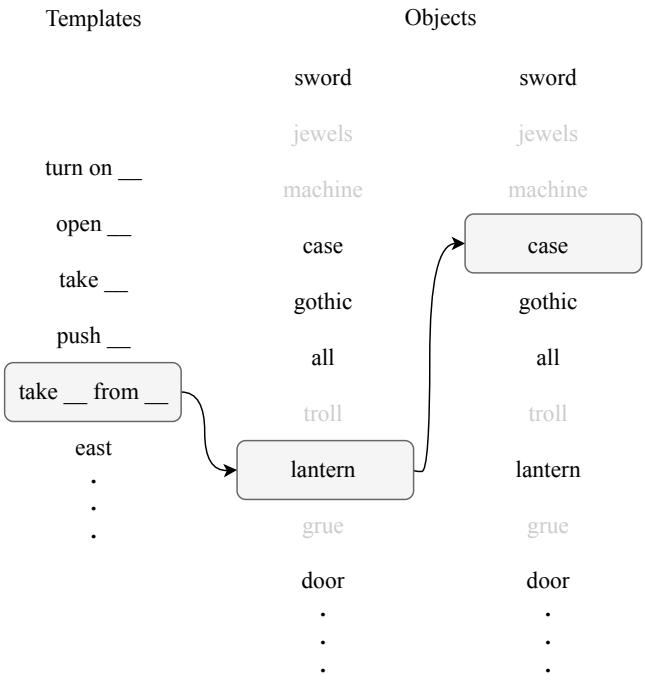


Figure 4.6: Visualization of the action decoding process using templates and objects. Objects consist of the entire game input vocabulary. Greyed out words indicate objects masked out by the knowledge graph.

mask, leveraging my knowledge graph at that timestep G_t to streamline the object decoding process. Formally, the *graph mask* $m_t = \{o : o \in G_t \wedge o \in V\}$, consists of all the entities found within the knowledge graph G_t and vocabulary V and is applied to the outputs of the object decoder GRUs—restricting them to predict objects in the mask. Generally, in an IF game, it is impossible to interact with an object that you never seen or that are not in your inventory and so the mask lets us explore the action space more efficiently. To account for cases where this assumption does not hold, i.e. when an object that the agent has never interacted with before must be referenced in order to progress in the game, I randomly add objects $o \in V$ to m_t with a probability p_m. An example of the graph-constrained action decoding process is illustrated in Fig. 4.6.

4.2.1 Training

I adapt the Advantage Actor Critic (A2C) method (Mnih *et al.* 2016) to train my network, using multiple workers to gather experiences from the simulator, making several significant changes along the way—as described below.

Valid Actions. Using a template-action space there are millions of possible actions at each step. Most of these actions do not make sense, are ungrammatical, etc. and an even fewer number of them actually cause the agent effect change in the world. Without any sense for which actions present valid interactions with the world, the combinatorial action space becomes prohibitively large for effective exploration.

I thus use the concept of *valid actions*, actions that can change the world in a particular state. These actions can usually be recognized through the game feedback, with responses like "Nothing happens" or "That phrase is not recognized." In practice, I follow Hausknecht *et al.* (2020) and use the valid action detection algorithm provided by Jericho. Formally, $Valid(s_t) = \{a_0, a_1 ... a_N\}$ and from this I can construct the corresponding set of *valid templates* $\mathcal{T}_{valid}(s_t) = \{\tau_0, \tau_1 ... \tau_N\}$. I further define a set of *valid objects* $\mathcal{O}_{valid}(s_t) = \{o_0, o_1 ... o_M\}$ which consists of all objects in the graph mask as defined in Sec. 4.2. This lets us introduce two cross-entropy loss terms to aid the action decoding process. The template loss given a particular state and current network parameters is applied to the decoder GRU_T. Similarly, the object loss is applied across the decoder GRU_O and is calculated by summing cross-entropy loss from all the object decoding steps.

$$\mathcal{L}_{\mathbb{T}}(s_t, a_t; \theta_t) = \frac{1}{N} \sum_{i=1}^{N} (y_{\tau_i} \log \pi_{\mathbb{T}}(\tau_i | s_t) + (1 - y_{\tau_i})(1 - \log \pi_{\mathbb{T}}(\tau_i | s_t)) \tag{4.12}$$

$$\mathcal{L}_{\mathbb{O}}(s_t, a_t; \theta_t) = \sum_{j=1}^{n} \frac{1}{M} \sum_{i=1}^{M} (y_{o_i} \log \pi_{\mathbb{O}_j}(o_i | s_t) + (1 - y_{o_i})(1 - \log \pi_{\mathbb{O}_j}(o_i | s_t))) \tag{4.13}$$

$$y_{\tau_i} = \begin{cases} 1 & \tau_i \in \mathcal{T}_{valid}(s_t) \\ 0 & else \end{cases} \quad y_{o_i} = \begin{cases} 1 & o_i \in \mathcal{O}_{valid}(s_t) \\ 0 & else \end{cases}$$

Updates. A2C training starts with calculating the advantage of taking an action in a state $A(s_t, a_t)$, defined as the value of taking an action $Q(s_t, a_t)$ compared to the average value of taking all possible *valid actions* in that state $V(s_t)$:

$$A(s_t, a_t) = Q(s_t, a_t) - V(s_t) \tag{4.14}$$

$$Q(s_t, a_t) = \mathbb{E}[r_t + \gamma V(s_{t+1})] \tag{4.15}$$

$V(s_t)$ is predicted by the critic as shown in Fig. 4.5 and r_t is the reward received at step t.

The action decoder or actor is then updated according to the gradient:

$$-\nabla_\theta(\log\pi_{\mathbb{T}}(\tau|s_t;\theta_t) + \sum_{i=1}^{n} \log\pi_{\mathbb{O}_i}(o_i|s_t, \tau, ..., o_{i-1};\theta_t))A(s_t, a_t) \tag{4.16}$$

updating the template policy $\pi_{\mathbb{T}}$ and object policies $\pi_{\mathbb{O}_i}$ based on the fact that each step in the action decoding process is conditioned on all the previously decoded portions. The critic is updated with respect to the gradient:

$$\frac{1}{2}\nabla_\theta(Q(s_t, a_t;\theta_t) - V(s_t;\theta_t))^2 \tag{4.17}$$

bringing the critic's prediction of the value of being in a state closer to its true underlying value. I further add an entropy loss over the valid actions, designed to prevent the agent from prematurely converging on a trajectory.

$$\mathcal{L}_{\mathbb{E}}(s_t, a_t;\theta_t) = \sum_{a \in V(s_t)} P(a|s_t)\log P(a|s_t) \tag{4.18}$$

4.2.2 Graph Ablations

The KG-A2C is tested on a suite of Jericho supported games and is compared to strong, established baselines. Additionally, as encouraged by Hausknecht *et al.* (2020), I present the set of handicaps used by my agents: (1) Jericho's ability to identify *valid actions* and (2) the Load, Save handicap in order to acquire $o_{t_{desc}}$ and $o_{t_{inv}}$ using the *look* and *inventory* commands without changing the game state.

Template DQN Baseline. I compare KG-A2C against Template-DQN, a strong baseline also utilizing the template based action space. TDQN (Hausknecht *et al.* 2020) is an extension of LSTM-DQN (Narasimhan *et al.* 2015) to template-based action spaces. This is accomplished using three output heads: one for estimating the Q-Values over templates $Q(s_t, u) \forall u \in \mathcal{T}$ and two for estimating Q-Values $Q(s_t, o_1), Q(s_t, o_2) \forall o_i \in \mathcal{O}$ over vocabulary to fill in the blanks of the template. The final executed action is constructed by greedily sampling from the predicted Q-values. Importantly, TDQN uses the same set of handicaps as KG-A2C allowing a fair comparison between these two algorithms.

Table 4.5 shows how KG-A2C fares across a diverse set of games supported by Jericho—testing the agent's ability to generalize to different genres, game structures, reward functions, and state-action spaces. KG-A2C matches or outperforms TDQN on 23 out of the 28 games that I test on. My agent is thus shown to be capable of extracting a knowledge graph that can sufficiently constrain the template based action space to enable effective exploration in a broad range of games.

In order to understand the contributions of different components of KG-A2C's architecture, I ablate KG-A2C's knowledge graph, template-action space, and valid-action loss. These ablations are performed on *Zork1*[2] and result in the following agents:

A2C. removes all components of KG-A2C's knowledge graph. In particular, the state embedding vector is now computed as $\mathbf{s_t} = \mathbf{o_t} \oplus \mathbf{c_t}$ and the *graph mask* is not used to

[2] A map of Zork1 with annotated rewards can be found in Appendix B along with a transcript of KG-A2C playing this game.

Table 4.5: Raw scores comparing KG-A2C (both with and without the mask) to TDQN across a wide set of games supported by Jericho. †Advent starts at a score of 36.

Game	$\|T\|$	$\|V\|$	**TDQN**	**KGA2C**	**KGA2C-unmasked**	**MaxRew**
905	82	296	0	0	0	1
acorncourt	151	343	**1.6**	0.3	0.3	30
advent†	189	786	36	36	36	350
adventureland	156	398	0	0	0	100
anchor	260	2257	0	0	0	100
awaken	159	505	0	0	0	50
balances	156	452	4.8	**10**	10	51
deephome	173	760	1	1	**29.2**	300
detective	197	344	169	**207.9**	141	360
dragon	177	1049	-5.3	**0**	-.2	25
enchanter	290	722	8.6	**12.1**	7.6	400
inhumane	141	409	0.7	3	**10.2**	300
jeIl	161	657	0	**1.8**	1.3	90
karn	161	657	**1.2**	0	0	90
library	173	510	6.3	**14.3**	9.6	30
ludicorp	187	503	6	17.8	**17.9**	150
moonlit	166	669	0	0	0	1
omniquest	207	460	**16.8**	3	5.4	50
pentari	155	472	17.4	**50.7**	50.4	70
snacktime	201	468	**9.7**	0	0	50
sorcerer	288	1013	5	5.8	**16.8**	400
spellbrkr	333	844	18.7	21.3	**30.1**	600
spirit	169	1112	0.6	**1.3**	**1.3**	250
temple	175	622	**7.9**	7.6	6.4	35
zenon	149	401	0	**3.9**	3.1	350
zork1	237	697	9.9	**34**	27	350
zork3	214	564	0	**.1**	.1	7
ztuu	186	607	4.9	**9.2**	5	100

constrain action decoding.

KG-A2C-no-gat. remove's the Graph Attention network, but retains the graph masking components. The knowledge graph is still constructed as usual but the agent uses the same state embedding vector as A2C.

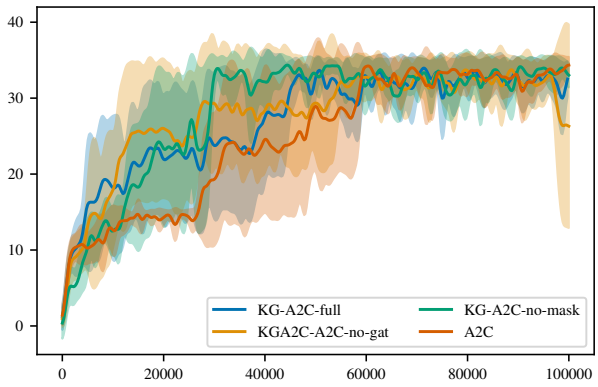
KG-A2C-no-mask. ablates the graph mask for purposes of action decoding. The knowledge graph is constructed as usual and the agent retains graph attention.

On *Zork1* as shown in Figure 4.7, I observe similar asymptotic performance between the all of the ablations – all reach approximately 34 points. This level of performance corresponds to a local optima where the agent collects the majority of available rewards without fighting the troll. Several other authors also report scores at this threshold (Zahavy *et al.* 2018; Jain *et al.* 2019). In terms of learning speed, the methods which have access to either the graph attention or the graph mask converge slightly faster than pure A2C which has neither.

To further understand these differences I performed a larger study across the full set of games comparing KG-A2C-full with KG-A2C-no-mask. The results in Table 4.5 show KG-A2C-full outperforms KG-A2C-no-mask on 10 games and is outperformed by KG-A2C-no-mask on 6. From this larger study I thus conclude the graph mask and knowledge graph are broadly useful components.

I perform two final ablations to study the importance of the supervised valid-action loss and the template action space:

KG-A2C-unsupervised. In order to understand the importance of training with valid-actions, KG-A2C-unsupervised is not allowed to access the list of *valid actions*—the valid-action-losses $\mathcal{L}_{\mathbb{T}}$ and $\mathcal{L}_{\mathbb{O}}$ are disabled and $\mathcal{L}_{\mathbb{E}}$ now based on the full action set. Thus, the agent must explore the template action space manually. KG-A2C-unsupervised, when trained for the same number of steps as all the other agents, fails to achieve any score. I can infer that the valid action auxiliary loss remains an important part of the overall algorithm, and access to the knowledge graph alone is not yet sufficient for removing this auxiliary



Agent	Mask	GAT
A2C		
KG-A2C-no-gat	✓	
KG-A2C-no-mask		✓
KG-A2C-full	✓	✓
KG-A2C-unsup	✓	✓

Figure 4.7: Ablation results on *Zork1*, averaged across 5 independent runs.

loss.

KG-A2C-seq. discards the template action space and instead decodes actions word by word up to a maximum of four words. A supervised cross-entropy-based valid action loss $\mathcal{L}_{\text{Valid}}$ is now calculated by selecting a random valid action $a_{t_{\text{valid}}} \in \text{Valid}(s_t)$ and using each token in it as a target label. As this action space is orders of magnitude larger than template actions, I use teacher-forcing to enable more effective exploration while training the agent—executing $a_{t_{\text{valid}}}$ with a probability $p_{\text{valid}} = 0.5$ and the decoded action otherwise. All other components remain the same as in the full KG-A2C.

KG-A2C-seq reaches a relatively low asymptotic performance of 8 points. This agent, using a action space consisting of the full vocabulary, performs significantly worse than the rest of the agents even when given the handicaps of teacher forcing and being allowed to train for significantly longer—indicating that the template based action space is also necessary for effective exploration.

4.3 On Policy: Q*BERT

Q*BERT is a also a reinforcement learning agent that uses a knowledge-graph to represent its understanding of the world state. Instead of using relation extraction rules as in KG-A2C, Q*BERT uses a variant of the BERT (Devlin *et al.* 2019) natural language trans-

former to answer questions and populate the knowledge graph from the answers.

Knowledge Graph State Representation I treat the problem of constructing the knowledge graph as a question-answering task. My method first extracts a set of graph vertices $\mathcal{V}$ by asking a question-answering system relevant questions and then linking them together using a set of relations $\mathcal{R}$ to form a knowledge graph representing information the agent has learned about the world. Examples of questions include: "What is my current location?", "What objects are around me?", and "What am I carrying?" to respectively extract information regarding the agent's current location, surrounding objects, inventory objects. Further, I predict attributes for each object by asking the question "What attributes does x object have?". An example of the knowledge graph that can be extracted from description text and the overall Q*BERT architecture are shown in Figure 4.8.

For question-answering, I use the pre-trained language model, ALBERT (Lan *et al.* 2020), a variant of BERT that is fine-tuned for question answering on the SQuAD 2.0 (Rajpurkar *et al.* 2018) question-answering dataset. I further fine-tune the ALBERT model on a dataset specific to the text-game domain, dubbed *Jericho-QA*.

4.3.1 Jericho-QA Dataset

The *Jericho-QA* dataset was created by making question answering pairs about text-games in the *Jericho* (Hausknecht *et al.* 2020)[3] framework as follows: For each game in Jericho, I use an oracle—an agent capable of playing the game perfectly using information normally off-limits such as the true game state—and a random exploration agent to gather ground truth state information about locations, objects, and attributes. From this ground truth, I construct pairs of questions in the form that Q*BERT will ask as it encounters environment description text, and the corresponding answers. These question-answer pairs are used to fine-tune the Q/A model and the ground truth data are discarded. No data from games I test Q*BERT on are used during ALBERT fine-tuning.

[3] https://github.com/microsoft/jericho

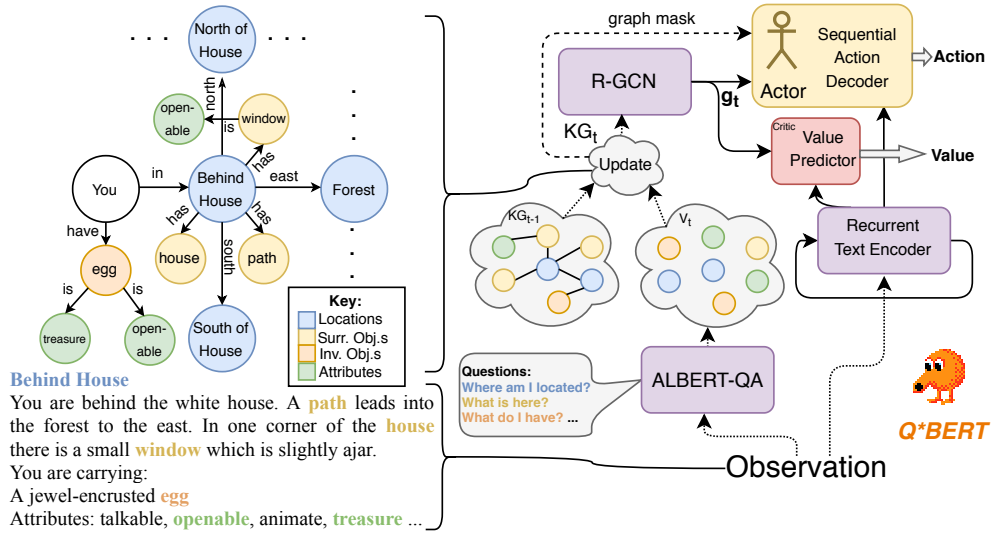


Figure 4.8: One-step knowledge graph extraction in the Jericho-QA format, and overall Q*BERT architecture at time step t. At each step the ALBERT-QA model extracts a relevant highlighted entity set V_t by answering questions based on the observation, which is used to update the knowledge graph.

Questions for QA were chosen on the basis of what past works in the area determined to be useful state information for the agent. For example, KG-A2C and GATA (Adhikari *et al.* 2020) explicitly differentiate between inventory/location descriptions/surrounding objects. The answers are annotated using information in the underlying world object tree that every text game is built on, this information can be accessed through the engine but, importantly, is used for annotation in the Jericho-QA dataset only. Jericho-QA data is formatted in the style of SQuAD 2.0 (Rajpurkar *et al.* 2018) and given samples of which questions are not applicable to certain states, i.e. negative samples.

Jericho-QA contains 221453 Question-Answer pairs in the training set and 56667 pairs in the held out test set. The test set consists of all the games that I test on in this chapter. The set of attributes for a game is taken directly from the game engine and is defined by the game developer.

A single sample looks like this:

```
Context:
[loc] Chief's Office You are standing in the chief's office. He is telling you, "The mayor
      was murdered yesterday night at 12:03 am. I want you to solve it before I get any bad
```

```
     publicity or the FBI has to come in." "Yessir!" you reply. He hands you a sheet of
    paper. once you have read it, go north or west. You can see a piece of white paper
    here.
[inv] You are carrying nothing.
[obs] [your score has just gone up by ten points.]
[atr] talkable, seen, lieable, enterable, nodwarf, indoors, visited, handed, lockable,
    surface, thing, water_room, unlock, lost, afflicted, is_treasure, converse, mentioned,
     male, npcworn, no_article, relevant, scored, queryable, town, pluggable, happy,
    is_followable, legible, multitude, burning, room, clothing, underneath, ward_area ,
    little, intact, animate, bled_in, supporter, readable, openable, near, nonlocal, door,
     plugged, sittable, toolbit, vehicle, light, lens_searchable, open, familiar,
    is_scroll, aimable, takeable, static, unique, concealed, vowelstart, alcoholic,
    bodypart, general, is_spell, full, dry_land, pushable, known, proper, inside, clean,
    ambiguously_plural, container, edible, treasure, can_plug, weapon, is_arrow,
    insubstantial, pluralname, transparent, is_coin, air_room, scenery, on, is_spell_book,
     burnt, burnable, auto_searched, locked, switchable, absent, rockable, beenunlocked,
    progressing, severed, worn, windy, stone, random, neuter, legible, female, asleep,
    wiped
Question: Where am I located? Answer: chief's office
Question: What is here? Answer: paper, west
Question: What do I have? Answer: nothing
Question: What attributes does paper have? Answer: legible, animate
Question: What attributes does west have? Answer: room, animate
```

4.3.2 Q*BERT Training

In a text-game the observation is a textual description of the environment. For every observation received, Q*BERT produces a fixed set of questions. The questions and the observation text are sent to the question-answering system. Predicted answers are converted into $\langle s, r, o \rangle$ triples and added to the knowledge graph. The complete knowledge graph is the input into Q*BERT's neural architecture (described below), which makes a prediction of the next action to take. At every step an observation consisting of several components is received: $o_t = (o_{t_{desc}}, o_{t_{game}}, o_{t_{inv}}, a_{t-1})$ corresponding to the room description, game feedback, inventory, and previous action, and total score R_t. The room description $o_{t_{desc}}$ is a textual description of the agent's location, obtained by executing the command "look". The game feedback $o_{t_{game}}$ is the simulators response to the agent's previous action and con-

sists of narrative and flavor text. The inventory $o_{t_{inv}}$ and previous action a_{t-1} components inform the agent about the contents of its inventory and the last action taken respectively.

Each of these components is processed using a GRU based encoder utilizing the hidden state from the previous step and combined to have a single observation embedding $\mathbf{o}_t$. At each step, I update my knowledge graph G_t using o_t as described in earlier in Section 4.3 and it is then embedded into a single vector $\mathbf{g_t}$. This encoding is based on the R-GCN and is calculated as:

$$\mathbf{g_t} = f\left(\mathbf{W_g}\sigma\left(\sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_i{}^r}\frac{1}{c_{i,r}}\mathbf{W_r}^{(l)}\mathbf{h}_j{}^{(l)} + \mathbf{W_0}^{(l)}\mathbf{h}_i{}^{(l)}\right) + \mathbf{b_g}\right) \tag{4.19}$$

Where $\mathcal{R}$ is the set of relations, $\mathcal{N}_i{}^r$ is the 1-step neighborhood of a vertex i with respect to relation r, $\mathbf{W_r}^{(l)}$ and $\mathbf{h}_j{}^{(l)}$ are the learnable convolutional filter weights with respect to relation r and hidden state of a vertex j in the last layer l of the R-GCN respectively, $c_{i,r}$ is a normalization constant, and $\mathbf{W_g}$ and $\mathbf{b_g}$ the weights and biases of the output linear layer. The full architecture can be found in Fig. 4.8. The state representation consists only of the textual observations and knowledge graph. Another key use of the knowledge graph, introduced as part of KG-A2C and described previously, is the *graph mask*, which restricts the possible set of entities that can be predicted to fill into the action templates at every step to those found in the agent's knowledge graph.

A2C training starts with calculating the advantage of taking an action in a state $A(s_t, a_t)$, defined as the value of taking an action $Q(s_t, a_t)$ compared to the average value of taking all possible *admissible actions* in that state $V(s_t)$:

$$A(s_t, a_t) = Q(s_t, a_t) - V(s_t) \tag{4.20}$$

$$Q(s_t, a_t) = \mathbb{E}[r_t + \gamma V(s_{t+1})] \tag{4.21}$$

The value is predicted by the critic as shown in Fig. 4.8 and r_t is the reward received at step t.

The action decoder or actor is then updated according to the gradient:

$$-\nabla_\theta(\log\pi_\mathbb{T}(\tau|s_t;\theta_t) + \sum_{i=1}^{n}\log\pi_{\mathbb{O}_i}(o_i|s_t,\tau,...,o_{i-1};\theta_t))A(s_t,a_t) \tag{4.22}$$

updating the template policy $\pi_\mathbb{T}$ and object policies $\pi_{\mathbb{O}_i}$ based on the fact that each step in the action decoding process is conditioned on all the previously decoded portions. The critic is updated with respect to the gradient:

$$\frac{1}{2}\nabla_\theta(Q(s_t,a_t;\theta_t) - V(s_t;\theta_t))^2 \tag{4.23}$$

bringing the critic's prediction of the value of being in a state closer to its true underlying value. An entropy loss is also added:

$$\mathcal{L}_\mathbb{E}(s_t,a_t;\theta_t) = \sum_{a\in V(s_t)} P(a|s_t)\log P(a|s_t) \tag{4.24}$$

4.3.3 Graph Evaluation

I evaluate the quality of the knowledge graph construction in a supervised setting. Next I perform an end-to-end evaluation in which knowledge graph construction is used by Q*BERT.

Table 4.7 shows the QA performance, and consequently the accuracy of the knowledge graphs built during exploration, on the Jericho-QA dataset using the rules-based approach of KG-A2C and the trained Albert-QA model in Q*BERT. Exact match (EM) is the percentage of times the model was able to predict the exact answer string, while F1 measures token overlap between prediction and ground truth. I observe a direct correlation between the quality of the extracted graph and an agent's performance on the games—Q*BERT in general possessing knowledge graphs of much higher quality than KG-A2C. On games where Q*BERT performed comparatively better than KG-A2C in terms of asymptotic

Table 4.6: Ground truth knowledge graph experiment results.

Expt.	**Q*BERT**		**MC!Q*BERT**
Game Reward	✓		✓
Intrinsic Motive			✓
Metric	**Eps.**	**Max**	**Max**
zork1	34.5	35	42
library	4.5	18	19
detective	246.1	288	338
balances	10	10	10
pentari	52.7	56	58
ztuu	5	5	12
ludicorp	18	19	23
deephome	1	1	6
temple	8	8	8

scores (columns 7 and 9), e.g. *detective*, the QA model had relatively high EM and F1, and vice versa as seen with *ztuu*. In general Q*BERT reaches comparable asymptotic performance to KG-A2C on 7 out of 9 games. However, as illustrated on *zork1* in Figure 4.9, Q*BERT reaches asymptotic performance faster than KG-A2C, indicating that the QA model improves learning; this trend is consistent on other games as shown in additional plots in Appendix B.2. Both agents rely on the graph to constrain the action space and provide a richer input state representation. Q*BERT uses a QA model fine-tuned on regularities of a text-game producing more relevant knowledge graphs than those extracted by OpenIE (Angeli *et al.* 2015) in KG-A2C for this purpose.

Further, in Table 4.6, I present results for the agents when given the ground truth knowledge graphs directly from the game engine. I see marginally greater performance across the board when compared to agents using constructed knowledge graphs (seen in Table 4.7). Q*BERT, when given a ground truth knowledge graph, shows matching or higher performance on 8 out of 9 games—with the sole exception being *library*.

Reinforcement learning offers an intuitive paradigm for exploring goal driven, contextually aware natural language generation. The sheer size of the natural language action space, However, has proven to be out of the reach of existing algorithms. In this chapter I introduced KG-DQN, KG-A2C, and Q*BERT—novel learning agents that demonstrates the feasibility of scaling reinforcement learning towards natural language actions spaces with

Table 4.7: QA results (EM and F1) on Jericho-QA test set and averaged asymptotic scores on games by different methods across 5 independent runs. For KG-A2C and Q*BERT, I present scores averaged across the final 100 episodes as well as *max scores*. Methods using exploration strategies show only *max scores* because Episode Average Score (*Eps.*) conflates forward progress and backtracking. Agents are allowed 10^6 steps for each parallel A2C agent with a batch size of 16.

Expt.	**QA Graph accuracy**				**Game reward**			
Agent	**KG-A2C**		**Q*BERT**		**KG-A2C**		**Q*BERT**	
Metric	**EM**	**F1**	**EM**	**F1**	**Eps.**	**Max**	**Eps.**	**Max**
zork1	6.08	8.42	43.93	48.31	34	35	34.1	35
library	10.33	26.74	49.78	52.76	14.3	19	10.0	18
detective	7.51	10.23	60.28	63.21	207.9	214	246.1	274
balances	32.53	36.09	85.81	86.18	10	10	10	10
pentari	16.48	23.36	65.02	69.54	50.7	56	51.2	56
ztuu	14.40	21.74	49.44	49.82	6	9	5	5
ludicorp	14.47	18.48	57.58	60.95	17.8	19	18	19
deephome	3.34	3.86	9.31	9.84	1	1	1	1
temple	7.42	9.44	48.98	49.17	7.6	8	7.9	8

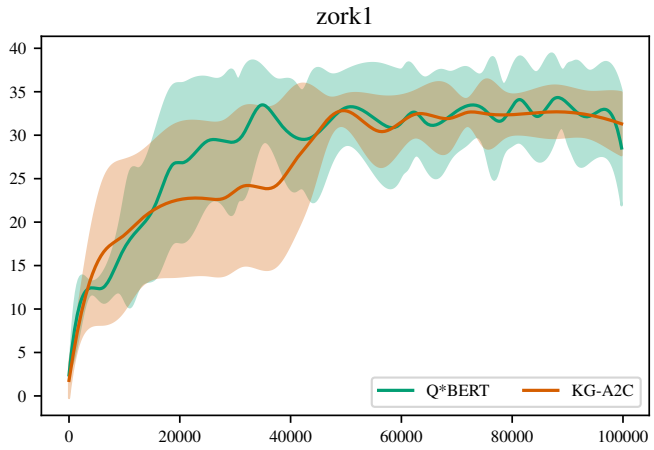


Figure 4.9: Episode rewards for KG-A2C and Q*BERT.

Figure 4.10: Select ablation results on *Zork1* conducted across 5 independent runs per experiment.

hundreds of millions of actions. The key insight to being able to efficiently explore such large spaces is the combination of a *knowledge-graph-based state space* and a *template-based action space*. The knowledge graph serves as a means for the agent to understand its surroundings, accumulate information about the game, and disambiguate similar textual observations while the template-based action space lends a measure of structure that enables us to exploit that same knowledge graph for language generation. Together they constrain the vast space of possible actions into the compact space of sensible ones. A suite of experiments across a diverse set of 28 human-made IF games shows wide improvement over TDQN, the current state-of-the-art template-based agent. Finally, an ablation study replicates state-of-the-art performance on *Zork1* even though Q*BERT is using an action space six orders of magnitude larger than previous agents—indicating the overall efficacy of my combined state-action space. The additional graph extraction results show that improving graph quality also improves sample efficiency of knowledge graph-based reinforcement learning agents.

CHAPTER 5
STRUCTURED EXPLORATION

Most text-adventure games are structured as quests with high branching factors in which players must solve a sequence of puzzles to advance the story and gain score—i.e. there are usually multiple ways to finish a quest. To solve these puzzles, players have freedom to a explore both new areas and previously unlocked areas of the game, collect clues, and acquire tools needed to solve the next puzzle and unlock the next portion of the game. From a Reinforcement Learning perspective, these puzzles can be viewed as bottlenecks that act as partitions between different regions of the state space. Whereas the multiple pathways to completion through puzzles may intuitively seem to make the problem easier, the opposite is true. I contend that existing Reinforcement Learning agents that are unaware of such latent structure and are thus poorly equipped for solving these types of problems.

In this chapter I introduce a new agent: MC!Q*BERT that builds on Q*BERT, designed with this latent structure in mind. As we saw in Chapter 4, Q*BERT improves on existing text-game agents that use knowledge graph-based state representations by framing knowledge graph construction during exploration as a question-answering task. To train Q*BERT's knowledge graph extractor, I introduced the *Jericho-QA* dataset for question-answering in text-games. I also showed that it leads to improved knowledge graph accuracy and sample efficiency compared to a rules-based approach.

However, improved knowledge graph accuracy is not enough to overcome bottlenecks; it does not improve asymptotic performance. To this end, MC!Q*BERT (Modular policy Chaining! Q*BERT) extends Q*BERT by combining two innovations: (1) an intrinsic motivation based on expansion of its knowledge graph both as a way to encourage exploration as well as a means for the agent to self-detect when it is stuck; and (2) by additionally introducing a structured exploration algorithm that, when stuck on a bottleneck, will back-

Observation: **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 here.

Action: **Open mailbox**

Observation: Opening the small mailbox reveals a leaflet.

Action: **Read leaflet**

Observation: (Taken) "WELCOME TO ZORK! ZORK is a game of adventure, danger, and low cunning. In it you will explore some of the most amazing territory ever seen by mortals. No computer should be without one!"

Action: **Go north**

Observation: **North of House** You are facing the north side of a white house. There is no door here, and all the windows are boarded up. To the north a narrow path winds through the trees.

Figure 5.1: Excerpt from *Zork1*.

track through the sequence of states leading to the current bottleneck, in search of alternative solutions. As MC!Q*BERT overcomes bottlenecks, it constructs a modular policy that chains together the solutions to multiple bottlenecks. Like Go Explore (Ecoffet *et al.* 2021), MC!Q*BERT relies on the determinism present in many text-games to reliably revisit previous states. However, I show that MC!Q*BERT's ability to detect bottlenecks via the knowledge graph state representation enable it to outperform such alternate exploration strategies on nine different games.

My contributions in this chapter are as follows: 1) I show that intrinsic motivation reward based on knowledge graph expansion is capable of reliably identifying bottleneck states. 2) Further, I show that structured exploration in the form of backtracking can be used to overcome these bottleneck states and reach state-of-the-art levels of performance on the Jericho benchmark (Hausknecht *et al.* 2020).

5.1 Understanding Bottleneck States

Overcoming bottlenecks is not as simple as selecting the correct action from the bottleneck state. Most bottlenecks have long-range dependencies that must first be satisfied: *Zork1* for instance features a bottleneck in which the agent must pass through the unlit *Cellar* where

a monster known as a Grue lurks, ready to eat unsuspecting players who enter without a light source. To pass this bottleneck the player must have previously acquired and lit the lantern. Other bottlenecks don't rely on inventory items and instead require the player to have satisfied an external condition such as visiting the reservoir control to drain water from a submerged room before being able to visit it. In both cases, the actions that fulfill dependencies of the bottleneck, e.g. acquiring the lantern or draining the room, are not rewarded by the game. Thus agents must correctly satisfy all *latent* dependencies, most of which are unrewarded, then take the right action from the correct location to overcome such bottlenecks. Consequently, most existing agents—regardless of whether they use a reduced action space (Zahavy *et al.* 2018; Yin and May 2019b) or the full space (Ammanabrolu and Hausknecht 2020; Hausknecht *et al.* 2020)—have failed to consistently clear these bottlenecks.

To better understand how to design algorithms that pass these bottlenecks, I first need to gain a sense for what they are. I observe that quests in text games can be modeled in the form of a dependency graph. These dependency graphs are directed acyclic graphs (DAGs) where the vertices indicate either rewards that can be collected or dependencies that must be met to progress and are generally unknown to a player *a priori*. In text-adventure games the dependencies are of two types: items that must be collected for future use, and locations that must be visited. An example of such a graph for the game of *Zork1* can found in Fig. 5.2.

More formally, bottleneck states are vertices in the dependency graph that, when the graph is laid out topographically, are (a) the only state on a level, and (b) there is another state at a higher level with non-zero reward. Bottlenecks can be mathematically expressed as follows: let $\mathcal{D} = \langle V, E \rangle$ be the directed acyclic dependency graph for a particular game where each vertex is tuple $v = \langle s_l, s_i, r(s) \rangle$ containing information on some state s such that s_l are location dependencies, s_i are inventory dependencies, and $r(s)$ is the reward associated with the state. There is a directed edge $e \in E$ between any two vertices such

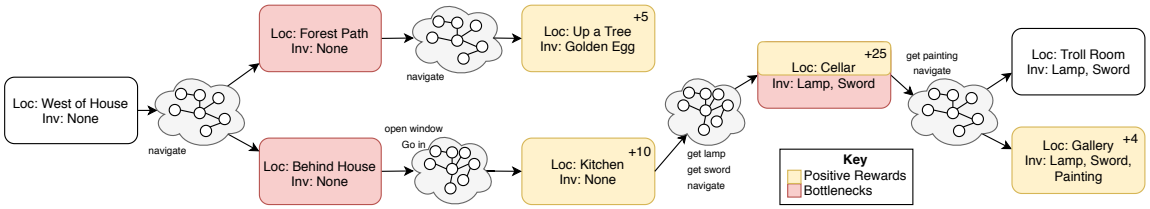


Figure 5.2: Portion of the *Zork1* quest structure visualized as a directed acyclic graph. Each node represents a state; clouds represent areas of high branching factor with labels indicating some of the actions that must be performed to progress

that the originating state meets the requirements s_l and s_i of the terminating vertex. $\mathcal{D}$ can be topologically sorted into levels $L = \{l_1, ..., l_n\}$ where each level represents a set of game states that are not dependant on each other. I formulate the set of all bottleneck states in the game:

$$\mathcal{B} = \{b : (|l_i| = 1, b \in l_i, V) \wedge (\exists s \in l_j \text{ s.t. } (j > i \wedge r(s) \neq 0))\} \tag{5.1}$$

This reads as the set of all states that that belong to a level with only one vertex and that there exists some state with a non-zero reward that depends on it. Intuitively, regardless of the path taken to get to a bottleneck state, any agent must pass it in order to continue collecting future rewards. *Behind House* is an example of a bottleneck state as seen in Fig. 5.2. The branching factor before and after this state is high but it is the only state through which one can enter the *Kitchen* through the window.

5.2 Structured Exploration

This section describes MC!Q*BERT an exploration method built on Q*BERT that detects overcomes bottlenecks by backtracking and policy chaining. This method of chaining policies and backtracking can be thought of in terms of *options* (Sutton *et al.* 1999; Stolle and Precup 2002), where the agent decomposes the task of solving the text game into the sub-tasks, each of which has it's own policy. In my case, each sub-task delivers the agent to a bottleneck state.

Algorithm 3 Structured Exploration

$\{\pi_{\text{chain}}, \pi_b, \pi\} \leftarrow \phi$ ▷ Chained, backtrack, current policy
$\{\mathcal{S}_b, \mathcal{S}\} \leftarrow \phi$ ▷ Backtrack, current state buffers
$s_0, r_{\text{init}} \leftarrow$ ENV.RESET()
$\mathcal{J}_{\text{max}} \leftarrow r_{\text{init}}, p \leftarrow 0$
for timestep t in 0...M **do** ▷ Train for M Steps
  $s_{t+1}, r_t, \pi \leftarrow$ Q*BERTUPDATE(s_t, π)
  $\mathcal{S} \leftarrow \mathcal{S} + s_{t+1}$ ▷ Append current state to state buffer
  $p \leftarrow p + 1$ ▷ Lose patience
  if $\mathcal{J}(\pi) \leq \mathcal{J}_{\text{max}}$ **then**
    if $p \geq patience$ **then** ▷ Stuck at a bottleneck
      $s_t, r_{\text{max}}, \pi \leftarrow$ BACKTRACK$(\pi_b, \mathcal{S}_b)$
      ▷ Bottleneck passed; Add π to the chained policy
      $\pi_{\text{chain}} \leftarrow \pi_{\text{chain}} + \pi$
  if $\mathcal{J}(\pi) > \mathcal{J}_{\text{max}}$ **then** ▷ New highscore found
    $\mathcal{J}_{\text{max}} \leftarrow \mathcal{J}(\pi); \pi_b \leftarrow \pi; \mathcal{S}_b \leftarrow \mathcal{S}; p \leftarrow 0$
return π_{chain} ▷ Chained policy that reaches max score

function Q*BERTUPDATE(s_t, π) ▷ One-step update
  $s_{t+1}, r_{g_t} \leftarrow$ ENV.STEP(s_t, π) ▷ Section 4.3
  $r_t \leftarrow$ CALCULATEREWARD(s_{t+1}, r_{g_t}) ▷ Eq. 5.3
  $\pi \leftarrow$ A2C.UPDATE(π, r_t) ▷ Appendix B.2
  return s_{t+1}, r_t, π

function BACKTRACK$(\pi_b, \mathcal{S}_b)$ ▷ Try to overcome bottleneck
  for b in REVERSE$(\mathcal{S}_b)$ **do** ▷ States leading to highscore
    $s_0 \leftarrow b; \pi \leftarrow \phi$
    for timestep t in 0...N **do** ▷ Train for N steps
      $s_{t+1}, r_t, \pi \leftarrow$ Q*BERTUPDATE(s_t, π)
      if $\mathcal{J}(\pi) > \mathcal{J}(\pi_b)$ **then return** s_t, r_t, π
  Terminate ▷ Can't find better score; Give up.

5.2.1 Bottleneck Detection using Intrinsic Motivation

Inspired by McGovern and Barto (2001), I present an intuitive way of detecting bottleneck states such as those in Fig. 5.2—or sub-tasks—in terms of whether or not the agent's ability to collect reward stagnates. If the agent does not collect a new reward for a number of environment interactions—defined in terms of a *patience* parameter—then it is possible that it is stuck due to a bottleneck state. An issue with this method, however, is that the placement of rewards does not always correspond to an agent being stuck. Complicating matters, rewards are sparse and often delayed; the agent not collecting a reward for a while might simply indicate that further exploration is required instead of truly being stuck.

To alleviate these issues, I define an *intrinsic motivation* for the agent that leverages the knowledge graph being built during exploration. The motivation is for the agent to learn more information regarding the world and expand the size of its knowledge graph. This provides us with a better indication of whether an agent is stuck or not—a stuck agent does not visit any new states, learns no new information about the world, and therefore does not expand its knowledge graph—leading to more effective bottleneck detection overall. To prevent the agent from discovering reward loops based on knowledge graph changes, I formally define this reward in terms of new information learned.

$$r_{\text{IM}_t} = \Delta(\mathcal{KG}_{\text{global}} - \mathcal{KG}_t) \text{ where } \mathcal{KG}_{\text{global}} = \bigcup_{i=1}^{t-1} \mathcal{KG}_i \tag{5.2}$$

Here $\mathcal{KG}_{\text{global}}$ is the set of all edges that the agent has ever had in its knowledge graph and the subtraction operator is a set difference. When the agent adds new edges to the graph perhaps as a the result of finding a new room $\mathcal{KG}_{\text{global}}$ changes and a positive reward is generated—this does not happen when that room is rediscovered in subsequent episodes. This is then scaled by the game score so the intrinsic motivation does not drown out the actual quest rewards, the overall reward the agent receives at time step t looks like this:

$$r_t = r_{g_t} + \alpha r_{\text{IM}_t} \frac{r_{g_t} + \epsilon}{r_{\text{max}}} \tag{5.3}$$

where ϵ is a small smoothing factor, α is a scaling factor, r_{g_t} is the game reward, r_{max} is the maximum score possible for that game, and r_t is the reward received by the agent on time step t.

5.2.2 Modular Policy Chaining

A primary reason that agents fail to pass bottlenecks is not satisfying all the required dependencies. To solve this problem, I introduce a method of policy chaining, where the agent uses the determinism of the simulator to backtrack to previously visited states in order to

fulfill dependencies required to overcome a bottleneck.

Specifically, Algorithm 3 optimizes the policy π as usual, but also keeps track of a buffer $\mathcal{S}$ of the distinct states and knowledge graphs that led up to each state (I use state s_t to colloquially refer to the combination of an observation o_t and knowledge graph $\mathcal{KG}_t$). Similarly, a bottleneck buffer $\mathcal{S}_b$ and policy π_b reflect the sequence of states and policy with the maximal return $\mathcal{J}_{\text{max}}$—consisting of the cumulative intrinsic as well as game rewards. A bottleneck is identified when the agents fails to improve upon $\mathcal{J}_{\text{max}}$ after *patience* number of steps, i.e. no improvement in raw game score or knowledge-graph-based intrinsic motivation reward. The agent then *backtracks* by searching backwards through the state sequence $\mathcal{S}_b$, restarting from each of the previous states—and training for N steps in search of a more optimal policy to overcome the bottleneck. When such a policy is found, it is appended to modular policy chain π_{chain}. Conversely, if no such policy is found, then I have failed to pass the current bottleneck and the training terminates.

5.3 Evaluation

I measure the utility of the knowledge graph-based intrinsic motivation in bottleneck detection and conduct an empirical comparison between MC!Q*BERT and other exploration strategies.

5.3.1 Intrinsic Motivation and Exploration Strategy Evaluation

MC!Q*BERT. Modularly Chained Q*BERT is evaluated by first testing policy chaining with only game reward and then with both game reward and intrinsic motivation. I provide a qualitative analysis of the bottlenecks detected with both methods with respect to those found in Fig. 5.2 on *Zork1*. Because MC!Q*BERT exploits structural domain assumptions that Q*BERT and KG-A2C cannot, I create a strong alternative baseline that looks at whether modular chaining improves over a related exploration strategy used in Go-Explore (Ecoffet *et al.* 2021).

Table 5.1: Averaged asymptotic scores on games by different methods across 5 independent runs. For KG-A2C and Q*BERT, I present scores averaged across the final 100 episodes as well as *max scores*. Methods using exploration strategies show only *max scores* because Episode Average Score (*Eps.*) conflates forward progress and backtracking. Agents are allowed 10^6 steps for each parallel A2C agent with a batch size of 16.

Expt.	**Game reward**					**Intrinsic**	
Agent	**KG-A2C**		**Q*BERT**		**MC!Q***	**MC!Q***	**GO!Q***
Metric	**Eps.**	**Max**	**Eps.**	**Max**	**Max**	**Max**	**Max**
zork1	34	35	34.1	35	32	41.6	31
library	14.3	19	10.0	18	19	19	18
detective	207.9	214	246.1	274	320	330	304
balances	10	10	10	10	10	10	10
pentari	50.7	56	51.2	56	56	58	40
ztuu	6	9	5	5	5	11.8	5
ludicorp	17.8	19	18	19	19	22.8	20.6
deephome	1	1	1	1	8	6	1
temple	7.6	8	7.9	8	8	8	8

GO!Q*BERT. is a baseline that makes the same underlying assumptions regarding the simulator as MC!Q*BERT but operates differently by tracking sub-optimal and under-explored states in order to allow the agent to explore upon more optimal states that may be a result of sparse rewards. This baseline trains Q*BERT in parallel to generate actions from the full action space used for exploration. It is based on the Go-Explore (Ecoffet *et al.* 2021) algorithm which consists of two phases, the first to continuously explore until a set of promising states and corresponding trajectories are found on the basis of total score, and the second to robustify this found policy against potential stochasticity in the game. Promising states are defined as those states when explored from will likely result in higher reward trajectories. Madotto *et al.* (2020) look at applying Go-Explore to text-games on a set of simpler games generated using the game generation framework TextWorld (Côté *et al.* 2018). They use a small set of "admissible actions"—actions guaranteed to change the world state at any given step during Phase 1—to explore and find high reward trajectories.

When MC!Q*BERT only uses game reward it matches Q*BERT on 5 out of 9 games and outperforms on 3 out of 9 games. When MC!Q*BERT uses intrinsic motivation plus game reward, it strictly outperforms KG-A2C and Q*BERT on 6 out of 9 games and matches it on the rest. MC!Q*BERT outperforms GO!Q*BERT on 7 games and matches

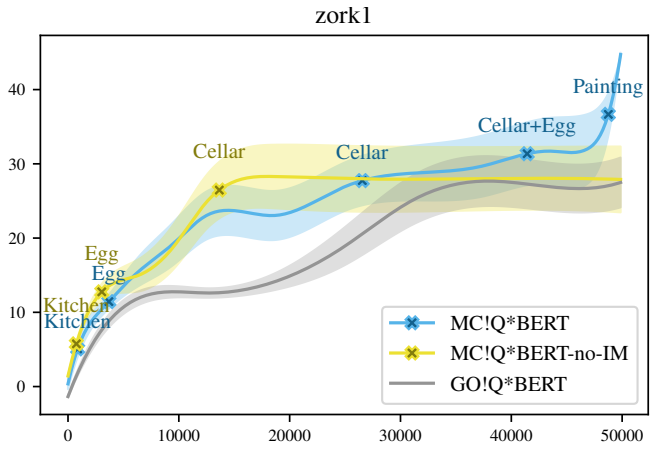


Figure 5.3: Max reward curves for exploration strategies.

Figure 5.4: Select ablation results on *Zork1* conducted across 5 independent runs per experiment. I see where the agents using structured exploration pass each bottleneck seen in Fig. 5.2. Q*BERT without IM is unable to detect nor surpass bottlenecks beyond the *Cellar*.

on 2, indicating that the modular chaining exploration strategy exploits the intrinsic motivation of knowledge graph learning better than the closest alternative exploration strategy.

5.4 Analysis

Table 5.1 shows that across all the games MC!Q*BERT matches or outperforms the current state-of-the-art when compared across the metric of the max score consistently received across runs. There are two main trends: First, MC!Q*BERT strongly benefits from the inclusion of intrinsic motivation rewards. Qualitatively, I illustrate this with *Zork1*, the canonical commercial text-adventure game that no RL agent has ever beaten. An analysis of bottlenecks detected by each agent in this game reveals differences in the overall accuracy of the bottleneck detection between MC!Q*BERT with and without intrinsic motivation. With intrinsic motivation, across 5 independent runs, MC!Q*BERT had an average true positive bottleneck state detection rate of 63%, false positive of 37%, with 50% coverage; and without it has a true positive rate of 58%, false positive of 42%, with coverage of

25%—assuming that the states such as in Fig. 5.2 represent the ground truth for bottlenecks. Coverage here refers to the number of unique bottlenecks states found during exploration compared to the total number of such states in the ground truth. This indicates that overall quality of bottleneck detection significantly improves given intrinsic motivation—enabling MC!Q*BERT to backtrack and surpass them. Figure 5.3 shows when each of these agents detect and subsequently overcome the bottlenecks outlined in Figure 5.2.

When intrinsic motivation is not used, the agent discovers that it can get to the *Kitchen* with a score of $+10$ and then *Cellar* with a score of $+25$ immediately after. It forgets how to get the *Egg* with a smaller score of $+5$ and never makes it past the Grue in the *Cellar*. Intrinsic motivation avoids this in two ways: (1) it makes it less focused on a locally high-reward trajectory—making it less greedy and helping it chain together rewards for the *Egg* and *Cellar*, and (2) provides rewards for fulfilling dependencies that would otherwise not be rewarded by the game—this is seen by the fact that it learns that picking up the lamp is the right way to surpass the *Cellar* bottleneck and reach the *Painting*. A similar behavior is observed with GO!Q*BERT: the agent settles prematurely on a locally high-reward trajectory and thus never has incentive to find more globally optimal trajectories by fulfilling the underlying dependency graph. Here, the likely cause is due to GO!Q*BERT's inability to backtrack and rethink discovered locally-maximal reward trajectories.

The results seen in Table 4.6, when the agent is given the ground truth knowledge graph, show that both Q*BERT and MC!Q*BERT perform on average better than when using graphs built from QA (or rules in the case of KG-A2C). This shows once again that knowledge graph accuracy is correlated to game performance, though the lower margin indicates that after a certain point—i.e. the accuracy levels of Q*BERT—gains in knowledge graph accuracy provide diminishing returns with respect to overall performance for this particular architecture.

Overall, we see that using both the improvements to graph construction in addition to intrinsic motivation and structured exploration consistently yields higher max scores across

a majority of the games when compared to the rest of the methods. Having just the improvements to graph building or structured exploration by themselves is not enough. Thus I infer that the full MC!Q*BERT agent is fundamentally exploring this combinatorially-sized space more effectively by virtue of being able to more consistently detect and clear bottlenecks. The improvement over systems using default exploration such as KG-A2C or Q*BERT by itself indicates that structured exploration is necessary when dealing with sparse and ill-placed reward functions.

5.5 Conclusions

Modern deep reinforcement learning agents using default exploration strategies such as ϵ-greedy are ill-equipped to deal with the latent structure of dependencies and bottlenecks found in many text-based games. To help address this challenge, I introduced two new agents: Q*BERT, an agent that constructs a knowledge graph of the world by asking questions about it, and MC!Q*BERT, which uses intrinsic motivation to grow the graph and detect bottlenecks arising from delayed rewards. A key insight from ablation studies is that the graph-based intrinsic motivation is crucial for bottleneck detection, preventing the agent from falling into locally optimal high reward trajectories due to ill-placed rewards. Policy chaining used in tandem with intrinsic motivation results in agents that explore further in the game by clearing bottlenecks more consistently.

I would like to conclude with a discussion on the relative differences in the assumptions that Q*BERT and MC!Q*BERT make regarding the underlying environment. Although both are framed as POMDPs, MC!Q*BERT makes stronger assumptions regarding the determinism of the game as compared to Q*BERT. MC!Q*BERT (and GO!Q*BERT) rely on the fact that the set of transition probabilities in a text-game are mostly deterministic. Using this, they are able to assume that frozen policies can be executed deterministically, i.e. with no significant deviations from the original trajectory. It is possible to robustify such policies by extending my method of structured exploration to perhaps perform im-

itation learning on the found highest score trajectories as seen in Phase 2 of the original GoExplore algorithm (Ecoffet *et al.* 2021). Stochasticity is not among set of challenges tackled in this work, however—I focus on learning how to better explore combinatorially-sized spaces with underlying long-term dependencies. For future works in this space, I believe that agents should be compared based on the set of assumptions made: agents like KG-A2C and Q*BERT when operating under standard reinforcement learning assumptions, and MC!Q*BERT and GO!Q*BERT when under the stronger assumption of having a deterministic simulator.

CHAPTER 6

COMMONSENSE REASONING IN TEXTUAL WORLDS

Many real-world activities can be thought of as a sequence of sub-goals in a partially observable environment. These activities—getting ready to go to work, for example—are considered trivial for humans because of *commonsense knowledge*. Commonsense knowledge is defined as a set of facts, beliefs, and procedures shared among many people in the same society or culture. However, to an agent learning purely by interacting with the environment, even simple tasks can require considerable trial-and-error.

Learning a control policy for a text-adventure game requires a significant amount of exploration, resulting in training runs that take hundreds of thousands of simulations (Ammanabrolu and Riedl 2019b; Narasimhan *et al.* 2015). One reason that text-adventure games require so much exploration is that most deep reinforcement learning algorithms are trained on a task without a real prior. In essence, the agent must learn everything about the game from only its interactions with the environment. Yet, text-adventure games make ample use of commonsense knowledge (e.g., an axe can be used to cut wood) and genre themes (e.g., in a horror or fantasy game, a coffin is likely to contain a vampire or other undead monster). This is in addition to the challenges innate to the text-adventure game itself—games are puzzles—which results in inefficient training.

I hypothesize that access to commonsense knowledge can enable an agent to more quickly converge on a policy that completes common, everyday tasks. I further hypothesize that commonsense knowledge can allow the agent to infer the presence of elements in the world when observations are noisy or fail. While some prior text-based game playing methods have incorporated commonsense knowledge (Murugesan *et al.* 2020; Fulda *et al.* 2017), I build off state of the art knowledge graph based techniques.

I explore the use of knowledge graphs and associated neural embeddings as a medium

for domain transfer to improve training effectiveness on new text-adventure games. Specifically, I explore transfer learning at multiple levels and across different dimensions. I first look at the effects of playing a text-adventure game given a strong prior in the form of a knowledge graph extracted from generalized textual walk-throughs of interactive fiction as well as those made specifically for a given game. Next, I explore the transfer of control policies in deep Q-learning (DQN) by pre-training portions of a deep Q-network using question-answering and by DQN-to-DQN parameter transfer between games. I evaluate these techniques on two different sets of human authored and computer generated games, demonstrating that my transfer learning methods enable us to learn a higher-quality control policy faster.

I further introduce two novel approaches that incorporate commonsense knowledge into game playing deep reinforcement learning agents using large scale pretraining, comparing my agents to the current state-of-the-art as a baseline. (1) I use a commonsense knowledge inference model to infer what can be known about the world based on text descriptions. Specifically, COMET (Bosselut *et al.* 2019) is a neural model that takes a simple sentence and infers what will be commonly believed about the people and objects referenced in the sentence. (2) Because commonsense knowledge also manifests itself as procedural knowledge, my final technique biases the agent toward sequences of action commands that BERT finds probable when predicting the next sentence. I further experiment with Q*BERT using a question-answering language model as source of commonsense knowledge.

6.1 Knowledge Graph Seeding

In this section I consider the problem of transferring a knowledge graph from a static text resource to a DQN—which I refer to as *seeding*. KG-DQN uses a knowledge graph as a state representation and also to prune the action space. This graph is built up over time, through the course of the agent's exploration. When the agent first starts the game, However, this graph is empty and does not help much in the action pruning process. The agent

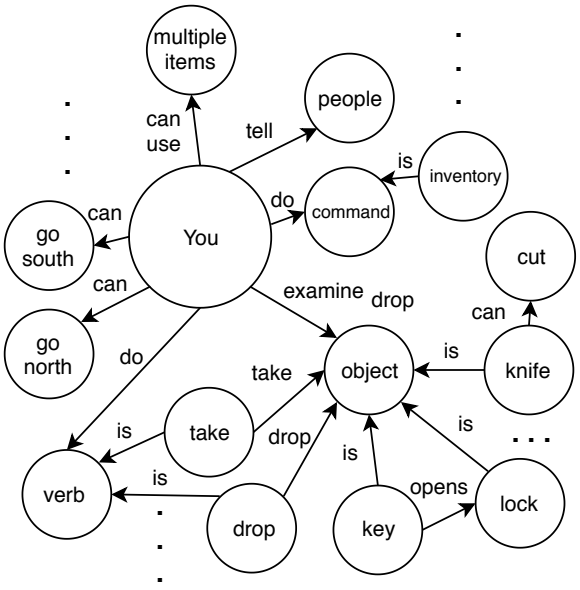


Figure 6.1: A select partial example of what a seed knowledge graph looks like. Ellipses indicate other similar entities and relations not shown.

thus wastes a large number of steps near the beginning of each game exploring ineffectively.

The intuition behind seeding the knowledge graph from another source is to give the agent a prior on which actions have a higher utility and thereby enabling more effective exploration. Text-adventure games typically belong to a particular genre of storytelling—e.g., horror, sci-fi, or soap opera—and an agent is at a distinct disadvantage if it doesn't have any genre knowledge. Thus, the goal of seeding is to give the agent a strong prior.

This seed knowledge graph is extracted from online general text-adventure guides as well as game/genre specific guides when available.[1] The graph is extracted from this the guide using a subset of the rules described in Chapter 3 used to extract information from the game observations, with the remainder of the RDF triples coming from OpenIE. There is no map of rooms in the environment that can be built, but it is possible to extract information regarding affordances of frequently occurring objects as well as common actions that can be performed across a wide range of text-adventure games. This extracted graph is thus potentially disjoint, containing only this generalizable information, in contrast to the graph extracted during the rest of the exploration process. An example of a graph used to seed

[1] An example of a guide I use is found here http://www.microheaven.com/IFGuide/step3.html

KG-DQN is given in Fig. 6.1. The KG-DQN is initialized with this knowledge graph.

6.2 Game Play as Question Answering

Previous work has shown that many NLP tasks can be framed as instances of question-answering and that in doing so, one can transfer knowledge between these tasks (McCann *et al.* 2017). In the abstract, an agent playing a text adventure game can be thought of as continuously asking the question "What is the right action to perform in this situation?" When appropriately trained, the agent may be able to answer the question for itself and select a good next move to execute. Treating the problem as question-answering will not replace the need for exploration in text-adventure games. However, I hypothesize that it will cut down on the amount of exploration needed during testing time, theoretically allowing it to complete quests faster; one of the challenges of text adventure games is that the quests are puzzles and even after training, execution of the policy requires a significant amount of exploration.

To teach the agent to answer the question of what action is best to take given an observation, I use an offline, pre-training approach. The data for the pre-training approach is generated using an oracle, an agent capable of finishing a game perfectly in the least number of steps possible. Specifically, the agent knows exactly what action to take given the state observation in order to advance the game in the most optimal manner possible. Through this process, I generate a set of traces consisting of state observations and actions such that the state observation provides the context for the implicit question of "What action should be taken?" and the oracle's correct action is the answer. I then use the DrQA (Chen *et al.* 2017) question-answering technique to train a paired question encoder and an answer encoder that together predict the answer (action) from the question (text observation). The weights from the SB-LSTM in the document encoder in the DrQA system are then used to initialize the weights of the SB-LSTM. Similarly, embedding layers of both the graph and the LSTM action encoder are initialized with the weights from the embedding layer of

same document encoder. Since the DrQA embedding layers are initialized with GloVe, I am transferring word embeddings that are tuned during the training of the QA architecture.

The game traces used to train the question-answering come from a set of games of the same domain but have different specific configurations of the environment and different quests. I use the TextWorld framework (Côté *et al.* 2018), which uses a grammar to generate random worlds and quests. The types of rooms are the same, but their relative spatial configuration, the types of objects, and the specific sequence of actions needed to complete the quest are different each time. This means that the agent cannot simply memorize quests. For pre-training to work, the agent must develop a general question-answering competence that can transfer to new quests. My approach to question-answering in the context of text adventure game playing thus represents a form of transfer learning.

6.3 Task Specific Transfer

The overarching goal of transfer learning in text-adventure games is to be able to train an agent on one game and use this training to improve the learning capabilities of another. There is growing body of work on improving training times on target tasks by transferring network parameters trained on source tasks (Yin H. and Pan 2017; Rusu *et al.* 2016; Rajendran *et al.* 2017). Of particular note is the work by Rusu *et al.* (2016), where they train a policy on a source task and then use this to help learn a new set of parameters on a target task. In this approach, decisions made during the training of the target task are jointly made using the frozen parameters of the transferred policy network as well as the current policy network.

My system first trains a question-answering system (Chen *et al.* 2017) using traces given by an oracle, as in Section 6.1. For commercial text-adventure games, these traces take the form of state-action pairs generated using perfect walkthrough descriptions of the game found online as described in Section 6.1.

I use the parameters of the question-answering system to pre-train portions of the deep

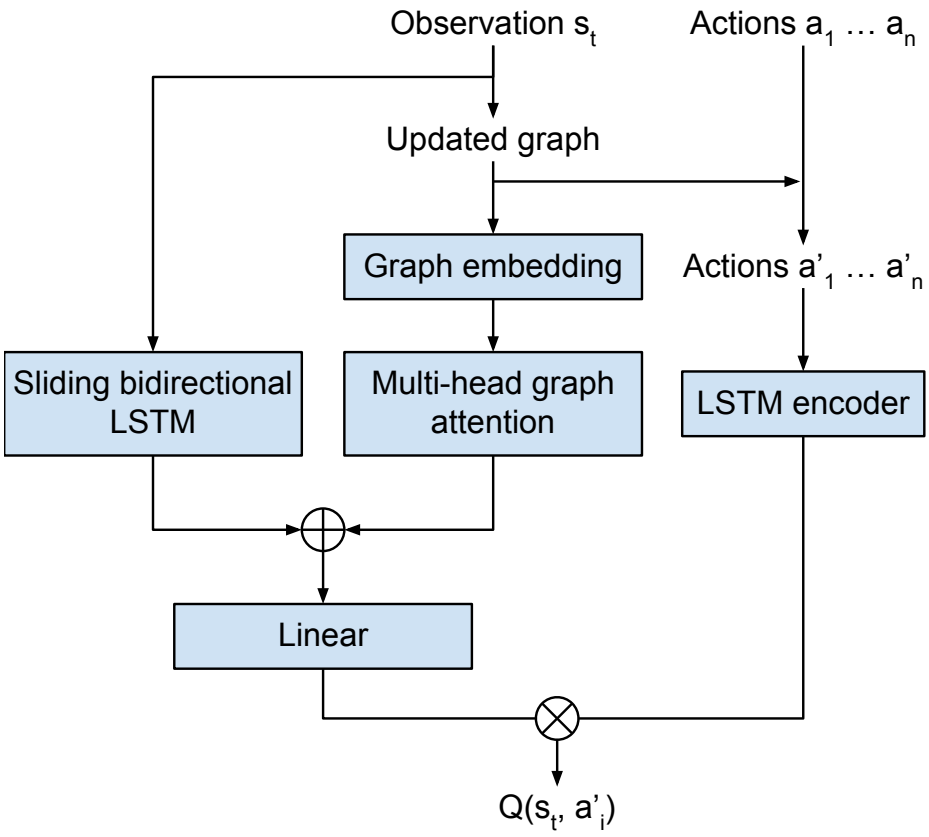


Figure 6.2: The KG-DQN transfer architecture.

Q-network for a different game within in the same domain. The portions that are pre-trained are the same parts of the architecture as in Ammanabrolu and Riedl (2019b). This game is referred to as the **source task**. The seeding of the knowledge graph is not strictly necessary but given that state-of-the-art DRL agents cannot complete real games, this makes the agent more effective at the source task.

I then transfer the knowledge and skills acquired from playing the source task to another game from the same genre—the **target task**. The parameters of the deep Q-network trained on the source game are used to initialize a new deep Q-network for the target task. All the weights indicated in the architecture of KG-DQN as shown in Fig. 6.2 are transferred. Unlike Rusu *et al.* (2016), I do not freeze the parameters of the deep Q-network trained on the source task nor use the two networks to jointly make decisions but instead just use it to initialize the parameters of the target task deep Q-network. This is done to account

for the fact that although graph embeddings can be transferred between games, the actual graph extracted from a game is non-transferable due to differences in structure between the games.

6.3.1 Evaluating Commonsense Transfer

I test my system on two separate sets of games in different domains using the Jericho and TextWorld frameworks (Hausknecht *et al.* 2020; Côté *et al.* 2018). The first set of games is "slice of life" themed and contains games that involve mundane tasks usually set in textual descriptions of normal houses. The second set of games is "horror" themed and contains noticeably more difficult games with a relatively larger vocabulary size and action set, non-standard fantasy names, etc. I choose these domains because of the availability of games in popular online gaming communities, the degree of vocabulary overlap within each theme, and overall structure of games in each theme. Specifically, there must be at least three games in each domain: at least one game to train the question-answering system on, and two more to train the parameters of the source and target task deep Q-networks. A summary of the statistics for the games is given in Table 6.1. Vocabulary overlap is calculated by measuring the percentage of overlap between a game's vocabulary and the domain's vocabulary, i.e. the union of the vocabularies for all the games I use within the domain. I observe that in both of these domains, the complexity of the game increases steadily from the game used for the question-answering system to the target and then source task games.

I perform ablation tests within each domain, mainly testing the effects of transfer from seeding, oracle-based question-answering, and source-to-target parameter transfer. Additionally, there are a couple of extra dimensions of ablations that I study, specific to each of the domains and explained below. All experiments are run three times using different random seeds. For all the experiments I report metrics known to be important for transfer learning tasks (Narasimhan *et al.* 2017; Taylor and Stone 2009): average reward collected

Table 6.1: Game statistics.

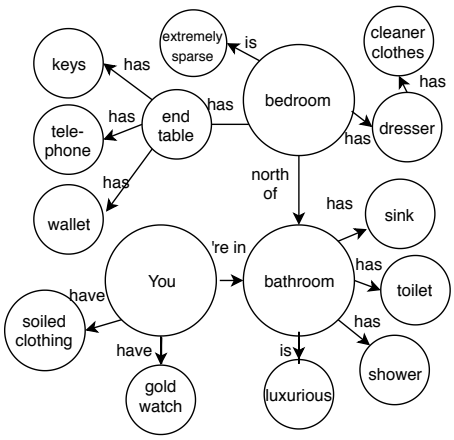
	Slice of life		Horror		
	QA/Source	Target	QA	Source	Target
	TextWorld	*9:05*	*Lurking Horror*	*Afflicted*	*Anchorhead*
Vocab size	788	297	773	761	2256
Branching factor	122	677	-	947	1918
Number of rooms	10	7	25	18	28
Completion steps	5	25	289	20	39
Words per obs.	65.1	45.2	68.1	81.2	114.2
New triples per obs.	6.4	4.1	-	12.6	17.0
% Vocab overlap	19.70	21.45	22.80	14.40	66.34
Max. aug. reward	5	27	-	21	43

in the first 50 episodes (init. reward), average reward collected for 50 episodes after convergence (final reward), and number of steps taken to finish the game for 50 episodes after convergence (steps). For the metrics tested after convergence, I set $\epsilon = 0.1$ following both Narasimhan *et al.* (2015) and Ammanabrolu and Riedl (2019b). I use similar hyperparameters to those reported in Ammanabrolu and Riedl (2019b) for training the KG-DQN with action pruning, with the main difference being that I use 100 dimensional word embeddings instead of 50 dimensions for the horror genre.

6.3.2 Slice of Life Experiments

TextWorld uses a grammar to generate similar games. Following Ammanabrolu and Riedl (2019b), I use TextWorld's "home" theme to generate the games for the question-answering system. TextWorld is a framework that uses a grammar to randomly generate game worlds and quests. This framework also gives us information such as instructions on how to finish the quest, and a list of actions that can be performed at each step based on the current world state. I do not let my agent access this additional solution information or admissible actions list. Given the relatively small quest length for TextWorld games—games can be completed in as little as 5 steps—I generate 50 such games and partition them into train and test sets in a 4:1 ratio. The traces are generated on the training set, and the question-answering system is evaluated on the test set.

I then pick a random game from the test set to train my source task deep Q-network



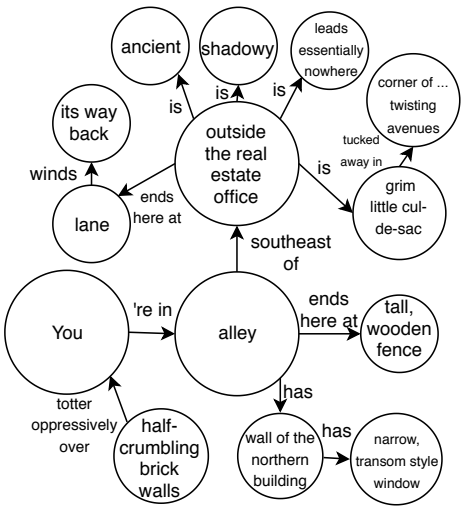
Bedroom
This bedroom is extremely spare, with dirty laundry scattered haphazardly all over the floor. Cleaner clothing can be found in the dresser. A bathroom lies to the south, while a door to the east leads to the living room. On the end table are a telephone, a wallet and some keys.
>*inventory*
You are carrying:
some soiled clothing (being worn)
a gold watch (being worn)
>*go south*
Bathroom
This is a far from luxurious but still quite functional bathroom, with a sink, toilet and shower. The bedroom lies to the north.

Figure 6.3: Partial unseeded knowledge graph example given observations and actions in the game *9:05*.

for this domain. For this training, I use the reward function provided by TextWorld: +1 for each action taken that moves the agent closer to finishing the quest; -1 for each action taken that extends the minimum number of steps needed to finish the quest from the current stage; 0 for all other situations.

I choose the game, *9:05*[2] as my target task game due to similarities in structure in addition to the vocabulary overlap. Note that there are multiple possible endings to this game and I pick the simplest one for the purpose of training my agent.

[2]https://ifdb.tads.org/viewgame?id=qzftg3j8nh5f34i2



Outside the Real Estate Office
A grim little cul-de-sac, tucked away in a corner of the claustrophobic tangle of narrow, twisting avenues that largely constitute the older portion of Anchorhead. Like most of the streets in this city, it is ancient, shadowy, and leads essentially nowhere. The lane ends here at the real estate agent's office, which lies to the east, and winds its way back toward the center of town to the west. A narrow, garbage-choked alley opens to the southeast.
>*go southeast*
Alley
This narrow aperture between two buildings is nearly blocked with piles of rotting cardboard boxes and overstuffed garbage cans. Ugly, half-crumbling brick walls to either side totter oppressively over you. The alley ends here at a tall, wooden fence. High up on the wall of the northern building there is a narrow, transom-style window.

Figure 6.4: Partial unseeded knowledge graph example given observations and actions in the game *Anchorhead*.

6.3.3 Horror Experiments

For the horror domain, I choose *Lurking Horror*[3] to train the question-answering system on. The source and target task games are chosen as *Afflicted*[4] and *Anchorhead*[5] respectively. However, due to the size and complexity of these two games some modifications to the games are required for the agent to be able to effectively solve them. I partition each of these games and make them smaller by reducing the final goal of the game to an intermediate checkpoint leading to it. This checkpoints were identified manually using walkthroughs of the game; each game has a natural intermediate goal. For example, *Anchorhead* is seg-

[3]https://ifdb.tads.org/viewgame?id=jhbd0kja1t57uop
[4]https://ifdb.tads.org/viewgame?id=epl4q2933rczoo9x
[5]https://ifdb.tads.org/viewgame?id=op0uw1gn1tjqmjt7

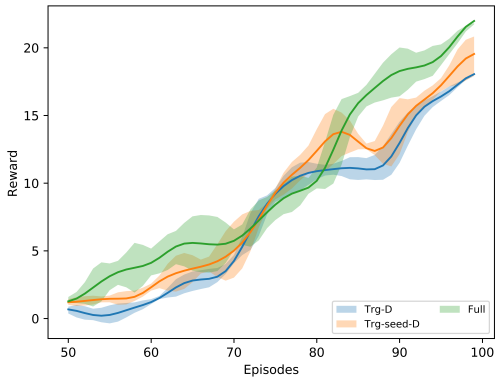


Figure 6.5: Reward curve for select experiments in the slice of life domain.

Table 6.2: Results for the slice of life games. "KG-DQN Full" refers to KG-DQN when seeded first, trained on the source, then transferred to the target. All experiment with QA indicate pre-training. S, D indicate sparse and dense reward respectively.

Experiment	Init. Rwd.	Final Rwd.	Steps
Source Game (*TextWorld*)			
KG-DQN no transfer	2.6 ± 0.73	4.7 ± 0.23	110.83 ± 4.92
KG-DQN w/ QA	2.8 ± 0.61	4.9 ± 0.09	88.57 ± 3.45
KG-DQN seeded	3.2 ± 0.57	4.8 ± 0.16	91.43 ± 1.89

mented into 3 chapters in the form of objectives spread across 3 days, of which I use only the first chapter. The exact details of the games after partitioning is described in Table 6.1. For Lurking Horror, I report numbers relevant for the oracle walkthrough. I then pre-prune the action space and use only the actions that are relevant for the sections of the game that I have partitioned out. The majority of the environment is still available for the agent to explore but the game ends upon completion of the chosen intermediate checkpoint.

6.3.4 Reward Augmentation

The combined state-action space for a commercial text-adventure game is quite large and the corresponding reward function is very sparse in comparison. The default, implied reward signal is to receive positive value upon completion of the game, and no reward value elsewhere. This is problematic from an experimentation perspective as text-adventure games are too complex for even state-of-the-art deep reinforcement learning agents to com-

plete. Even using transfer learning methods, a sparse reward signal usually results in ineffective exploration by the agent.

To make experimentation feasible, I augment the reward to give the agent a dense reward signal. Specifically, I use an oracle to generate state-action traces (identical to how as when training the question-answering system). An oracle is an agent that is capable of playing and finishing a game perfectly in the least number of steps possible. The state-action pairs generated using perfect walkthroughs of the game are then used as checkpoints and used to give the agent additional reward. If the agent encounters any of these state-action pairs when training, i.e. performs the right action given a corresponding state, it receives a proportional reward in addition to the standard reward built into the game. This reward is scaled based on the game and is designed to be less than the smallest reward given by the original reward function to prevent it from overpowering the built-in reward. I refer to agents using this technique as having "dense" reward and "sparse" reward otherwise. The agent otherwise receives no information from the oracle about how to win the game.

The structure of the experiments are such that the for each of the domains, the target task game is more complex that the source task game. The slice of life games are also generally less complex than the horror games; they have a simpler vocabulary and a more linear quest structure. Additionally, given the nature of interactive fiction games, it is nearly impossible—even for human players—to achieve completion in the minimum number of steps (as given by the steps to completion in Table 6.1); each of these games are puzzle based and require extensive exploration and interaction with various objects in the environment to complete.

Table 6.2 and Table 6.3 show results for the slice of life and horror domains, respectively. In both domains seeding and QA pre-training improve performance by similar amounts from the baseline on both the source and target task games. A series of t-tests comparing the results of the pre-training and graph seeding with the baseline KG-DQN show that all results are significant with $p < 0.05$. Both the pre-training and graph seeding

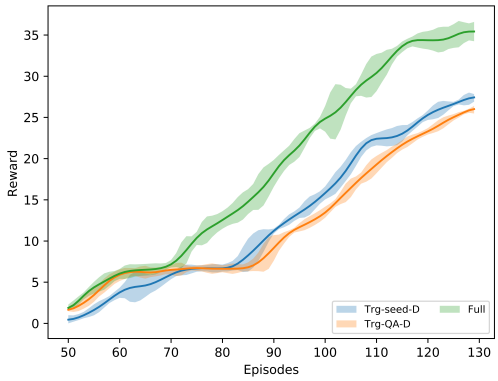


Figure 6.6: Reward curve for select experiments in the horror domain.

Table 6.3: Results for horror games. Note that the reward type is dense for all results. "KG-DQN Full" refers to KG-DQN seeded, transferred from source. All experiment with QA indicate pre-training.

Experiment	Init. Rwd.	Final Rwd.	Steps
Source Game (*Afflicted*)			
KG-DQN no transfer	3.0 ± 1.3	14.1 ± 1.73	1934.7 ± 85.67
KG-DQN w/ QA	4.3 ± 1.34	15.1 ± 1.60	1179 ± 32.07
KG-DQN seeded	4.1 ± 1.19	14.6 ± 1.26	1125.3 ± 49.57
Target Game (*Anchorhead*)			
KG-DQN untuned	-	3.8 ± 0.23	-
KG-DQN no transfer	1.0 ± 0.34	6.8 ± 0.42	-
KG-DQN w/ QA	3.6 ± 0.91	24.8 ± 0.6	4874 ± 90.74
KG-DQN seeded	1.7 ± 0.62	26.6 ± 0.42	4937 ± 42.93
KG-DQN full	4.1 ± 0.9	39.9 ± 0.53	4334.3 ± 56.13

perform similar functions in enabling the agent to explore more effectively while picking high utility actions.

Even when untuned, i.e. evaluating the agent on the target task after having only trained on the source task, the agent shows better performance than training on the target task from scratch using the sparse reward. As expected, we see a further gain in performance when the dense reward function is used for both of these domains as well. In the horror domain, the agent fails to converge to a state where it is capable of finishing the game without the dense reward function due to the horror games being more complex.

When an agent is trained using on just the target task horror game, *Anchorhead*, it does not converge to completion and only gets as far as achieving a reward of approximately

7 (max. observed reward from the best model is 41). This corresponds to a point in the game where the player is required to use a term in an action that the player has never observed before, "look up Verlac" when in front of a certain file cabinet—"Verlac" being the unknown entity. Without seeding or QA pre-training, the agent is unable to cut down the action space enough to effectively explore and find the solution to progress further. The relative effectiveness of the gains in initial reward due to seeding appears to depend on the game and the corresponding static text document. In all situations except Anchohead, seeding provides comparable gains in initial reward as compared to QA — there is no statistical difference between the two when performing similar t-tests.

When the full system is used—i.e. I seed the knowledge graph, pre-train QA, then train the source task game, then the target task game using the augmented reward function—I see a significant gain in performance, up to an 80% gain in terms of completion steps in some cases. The bottleneck at reward 7 is still difficult to pass, however, as seen in Fig. 6.6, in which I can see that the agent spends a relatively long time around this reward level unless the full transfer technique is used. I further see in Figures 6.5, 6.6 that transferring knowledge results in the agent learning this higher quality policy much faster. In fact, I note that training a full system is more efficient than just training the agent on a single task, i.e. training a QA system then a source task game for 50 episodes then transferring and training a seeded target task game for 50 episodes is more effective than just training the target task game by itself for even 150+ episodes.

I have demonstrated that using knowledge graphs as a state representation enables efficient transfer between deep reinforcement learning agents designed to play text-adventure games, reducing training times and increasing the quality of the learned control policy. My results show that I are able to extract a graph from a general static text resource and use that to give the agent knowledge regarding domain specific vocabulary, object affordances, etc. Additionally, I demonstrate that I can effectively transfer knowledge using deep Q-network parameter weights, either by pre-training portions of the network using a

question-answering system or by transferring parameters from a source to a target game. My agent trains faster overall, including the number of episodes required to pre-train and train on a source task, and performs up to 80% better on convergence than an agent not utilizing these techniques.

I conclude that knowledge graphs enable transfer in deep reinforcement learning agents by providing the agent with a more explicit–and interpretable–mapping between the state and action spaces of different games. This mapping helps overcome the challenges twin challenges of partial observability and combinatorially large action spaces inherent in all text-adventure games by allowing the agent to better explore the state-action space.

6.4 Commonsense via Large Scale Pre-training

I experiment with three agents, each with their own approach for incorporating commonsense knowledge to augment policy learning. All three agents build off the KG-A2C (Ammanabrolu and Hausknecht 2020) agent framework, which is shown in Figure 6.7. At every step, KG-A2C uses a heuristic information extraction process to identify $\langle$ *subject, relation, object* $\rangle$ triples in the current room's text description. These triples are added to an ever-growing knowledge graph, which is embedded and used to inform the choice of action (text command). The knowledge graph is the agent's belief about the state of the world experienced to date. KG-A2C filters out actions that contain object references not contained in the graph.

The COMET-A2C Agent. This agent is similar to Q*BERT but replaces ALBERT with COMET Bosselut *et al.* (2019), a neural commonsense inference model (Figure 6.7). I use the version of COMET trained on the ConceptNet (Speer and Havasi 2012) dataset to take text sentences and generate a number of short phrases that may be inferred from the input text. COMET produces several types of inference templates. I specifically use COMET's *HasA* inference class. COMET-A2C uses KG-A2C's information extraction process to produce $\langle subject, relation, object \rangle$ triples; relationships inferred by COMET

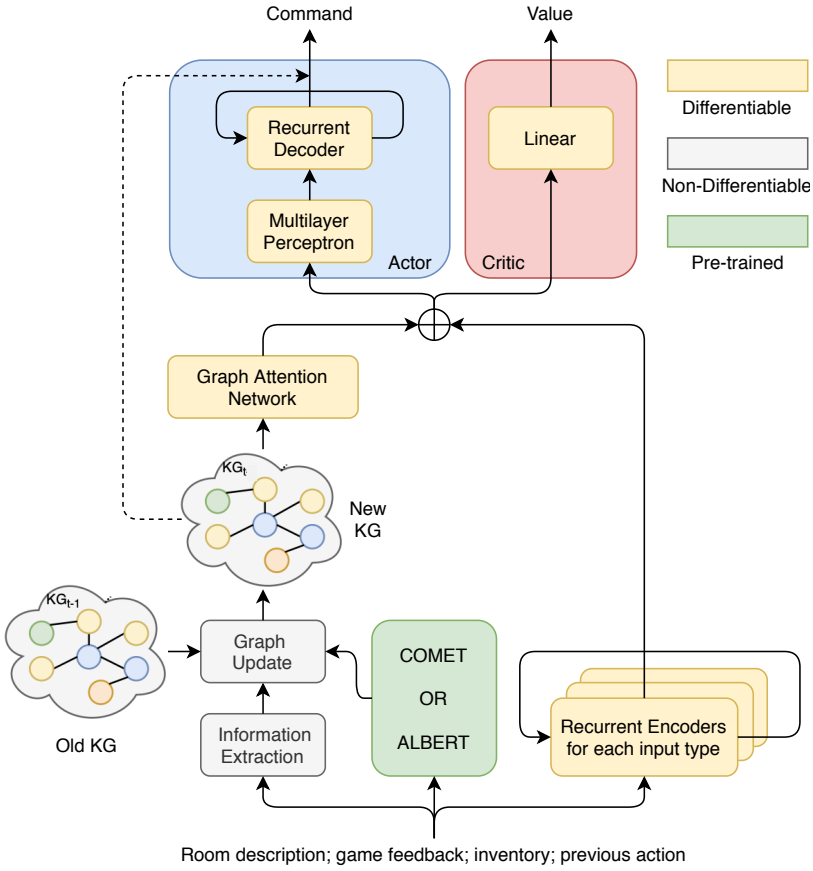


Figure 6.7: Augmented KG-A2C architecture.

are then added. I hypothesize that COMET will make the agent's understanding of the world state more robust by inferring the existence of objects commonly found in certain types of locations. Figure 6.8 shows an example of the knowledge graph generated by COMET-A2C for a given text description.

The KG-A2C-BERT Agent. This agent is identical to KG-A2C, except that it uses a policy-shaping method for exploration (Griffith 2018). Policy-shaping is a technique whereby an external source of knowledge is used to re-rank a distribution over output actions during training. KG-A2C-BERT samples the top k action commands generated by the network and scores each based on a history of previous commands. This is done by concatenating the currently proposed command to prior commands and use BERT to compute $P_\theta(c_t|c_1...c_{t-1})$ where c_i is a command at time step t and θ is BERT's pre-trained weights. The k candidate commands are re-ranked according to the score and the agent re-samples

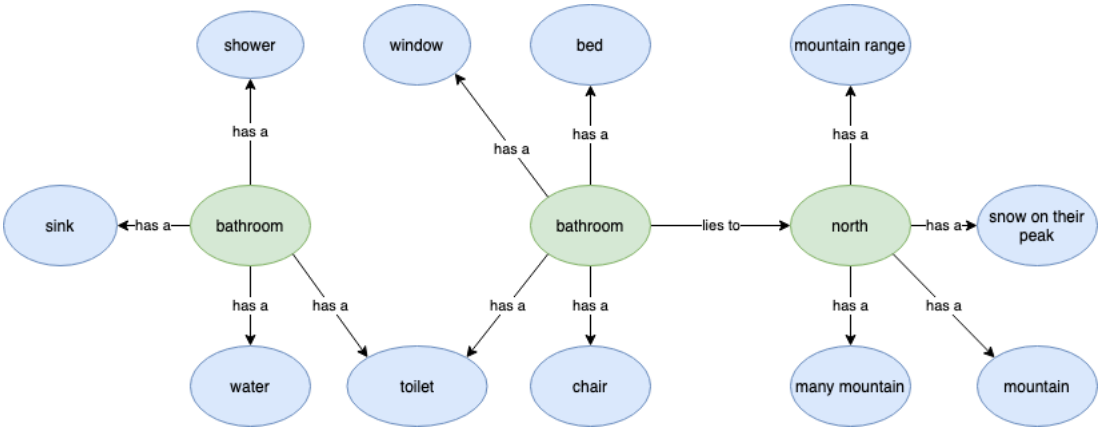


Figure 6.8: Knowledge Graph generated by COMET-A2C for the observation: *This is a far from luxurious but still quite functional bathroom. The bedroom lies to the north.*

Original observation

Bathroom
This is a far from luxurious but still quite functional bathroom, with a sink, toilet and shower. The bedroom lies to the north.

Modified Observation

Bathroom
This is a far from luxurious but still quite functional bathroom. The bedroom lies to the north.

Figure 6.9: The original and modified observation for the *Bathroom* in *9:05*.

from the new distribution. In Figure 6.7 the green box is removed and re-ranking is applied to the output of the actor module.

The Q*BERT Agent. Q*BERT(4), which augments the knowledge graph by using the pre-trained question-answering model, ALBERT (Lan *et al.* 2020). ALBERT is first fine-tuned on a dataset specific to the text-game domain. Q*BERT generates questions about the current room environment and ALBERT's answers are converted into $\langle subject, relation, object \rangle$ and added to the knowledge graph (Figure 6.7). While Q*BERT was not explicitly design with commonsense knowledge in mind, I hypothesize that ALBERT can extrapolate from room text description using knowledge acquired through training on a large corpus of texts.

6.4.1 Evaluating Commonsense Pre-training

I conduct experiments in the *9:05* slice of life text-based game. In this game, the player must get ready for work by taking a shower, wearing clean clothes and then travel to the workplace by car. The game provides a single reward of 1 or 0 at the end. The branching factor is very high and the only reward requires 25-30 steps executed in the perfect order. Due to the extreme sparseness of this feedback, all agents struggle to make any significant progress. Consequently, I provide a shaped reward function to add reward density. The agent is rewarded +1 for each of six different actions necessary to complete the task of taking a shower (Appendix A3). These states may only be observed in sequence and loops cannot occur.

I conducted two experiments. (1) I assess performance in a version of *9:05* where reward is given for passing key states. (2) I test agents' performance when required to supplement missing/failed observations with commonsense inferences. A modified version of *9:05* has the shaped reward but also deletes all textual references to three critical objects; the `sink`, `toilet`, and `shower` are omitted from the `bathroom` description. See Figure 6.9 This approximates situations where the agent's observations may have failed to observe the objects, or to correctly parse and extract relations pertaining to these objects. It also simulates the way in which humans recognize that it is unnecessary or obvious to state facts that everyone would likely agree upon. The objects were not removed, only their mentions in the text descriptions.

6.4.2 Results and Analysis

Experiment 1 results are shown in Figure 6.10 (left), which plots reward per time step averaged over five runs. The solid lines are the smoothed mean reward and the shaded areas show one standard deviation of reward values. KG-A2C gets stuck after entering the bathroom and never makes it to the shower. KG-A2C-BERT can make it to reward 5 but does so unreliably—well outside the standard deviation. Since this is rarely achieved, the

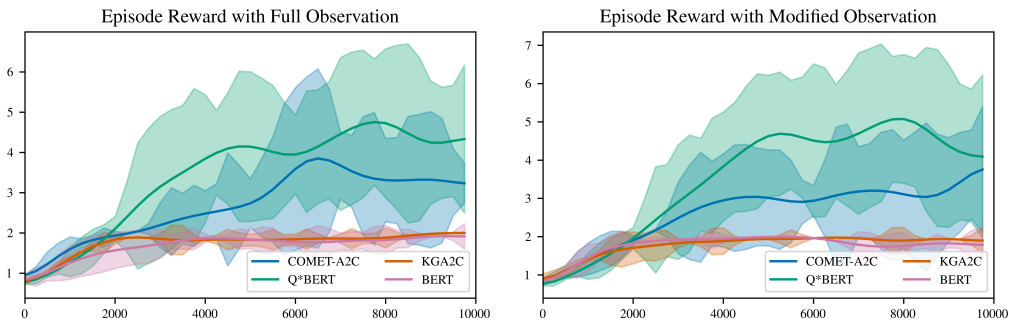


Figure 6.10: Reward performance for all agents on *9:05* with full observations (left) and modified observations (right). The solid lines show a smoothed average performance with standard deviation over 5 independent runs.

average performance is similar to that of KG-A2C.

COMET-A2C and Q*BERT are both able to get past the shower (reward 6) and to the next phase of the game where player drives to work. Their performances in this experiment are not significantly different, although Q*BERT achieves reward 6 or greater more frequently and thus has a higher mean reward.

Experiment 1 validates that commonsense knowledge helps agent performance in *9:05*, which makes heavy use of locations and situations that also commonly occur in the real world. KG-A2C-BERT performs better than KG-A2C because BERT informs the agent's exploration by comparing action command sequences to patterns BERT recognizes. However, exploration is stochastic and it rarely progresses far into the game. COMET-A2C adds *HasA* relations to the knowledge graph and this helps correlate a richer state with the best action to take. Q*BERT likely performs better to COMET-A2C due to richer, more diverse correlations between state and action; this may help navigate actions between the key rewarded steps.

The results for Experiment 2 are shown in Figure 6.10 (right). In this experiment, agents contend with missing object references in room descriptions. KG-A2C never makes it past a score of 2. It enters the bathroom but cannot complete any tasks due to the inability to directly observe the sink, toilet, or shower. KG-A2C-BERT's performance is identical to KG-A2C because action commands are filtered out that do not reference objects in the

knowledge graph. Whereas filtering normally reduces unlikely exploration, it hurts when the agent fails to observe objects or observations are unreliable.

COMET-A2C and Q*BERT are able to use the sink, toilet, and shower to successfully complete all the tasks required in the bathroom which leads to greater reward. As with Experiment 1, both COMET-A2C and Q*BERT are able to occasionally progress beyond the bathroom—beyond reward 6. As before, Q*BERT has a non-significantly higher average reward because it more consistently passes the shower task, whereas COMET-A2C has more variance in performance.

Experiment 2 confirms my intuitions about the role that commonsense inferences are playing in the agent's decision-making. By making the presence of key objects in a location implicit instead of explicit, I verify in a controlled fashion that commonsense inferences beneficially augments agents' senses. The difference in performance between Q*BERT and COMET-A2C is due to the way they infer commonsense information as detailed earlier. Both infer the existence of the missing entities, allowing them to progress through the game.

It is natural for commonsense details to be omitted in natural language. This work demonstrates a deep reinforcement learning framework for "acting through language" can be made more robust to real-world natural language phenomena such as assumptions that an interlocutor shares commonsense knowledge. To that end, experiments with agents that can conduct activities while interacting with common natural language phenomena may lead to agents that are better at interacting with humans.

I conducted experiments in slice of life text-based games to understand how commonsense knowledge can help agents handle puzzles which are simulacrum of real world locations and scenarios. Slice of life games, and *9:05* in particular, require the player to recreate behavior that might be also conducted routinely in the real-world. My experiments show that commonsense inferences—regardless of the source of commonsense inference—can be used to augment an agent's beliefs about the state of the world, making the agent more robust to failed observations or intentional omissions in environment descriptions.

CHAPTER 7

LEARNING TO BALANCE ACTIONS AND DIALOGUE

This chapter builds on LIGHT (Urbanek *et al.* 2019), a large-scale crowdsourced fantasy text-adventure game, consisting of a set of locations, characters, and objects possesses rich textual worlds. On top of the other text-game related challenges tackled earlier, the primary core challenge for the agent here is to operate in spaces where the agent can both act as well as speak to other characters using dialogue. This requires the recognition that dialogue can also be used to change the environment. With dialogue, an agent can now learn to instruct or convince other characters in the world to achieve the goal for it—e.g. convince the pirate through dialogue to give you their treasure instead of just stealing it yourself. The agent needs to learn to balance both its ability to speak as well as act in order to effectively achieve its goals (Ammanabrolu *et al.* 2021).

LIGHT contains rich descriptions of textual worlds without any notion of goals to train goal-driven agents. I present a dataset of quests for LIGHT and demonstrations of humans playing these quests (as seen in Figures 7.2 and 7.3), providing natural language descriptions in varying levels of abstraction of motivations for a given character in a particular setting. In the example in Figure 7.2, a potential short term motivation for a dragon who has lost its golden egg is to recover it, but the underlying goal in the long run is to build as large a treasure hoard as possible.

To complete these quests, an agent must reason about potential actions and utterances based on incomplete descriptions of the locations, objects, and other characters. When a human is placed in a fantasy setting such as LIGHT, they already know that kings are royalty and must be treated respectfully, swords are weapons, etc.—commonsense knowledge that a learning agent must acquire to ensure successful interactions. To equip agents with relevant priors in such worlds, I built a new large-scale knowledge base by domain-adapting

the large-scale commonsense knowledge graph ATOMIC (Sap *et al.* 2019) to the LIGHT fantasy world. This knowledge base was dubbed ATOMIC-LIGHT.

Agents in these worlds need to speak naturally and act consistently with respect to their motivations in order to make progress towards achieving the goals outlined by them. I then introduce a reinforcement learning (RL) system that incorporates large-scale language modeling and the above commonsense-based pre-training. I show that RL is superior to behavior cloning or other supervised training on our data; and that carefully combining pre-training with RL is superior to either.

However, I find that although pre-training can be an effective tool in this setting, it requires more finesse than in the standard supervised setting. In particular, I find that simply pre-training a model on a large "generic" corpus (Sap *et al.* 2019; Baumgartner *et al.* 2020) of commonsense/language data or pre-training on the domain specific LIGHT corpus, and then fine-tuning via RL is *less* effective than training RL from scratch. Furthermore, by carefully combining general and domain-specific pre-training, I observe large improvements over RL from scratch.

In short, the contributions of this chapter are threefold: (1) A dataset of quests, LIGHT-Quests, and a companion fantasy themed commonsense knowledge graph ATOMIC-LIGHT; (2) a reinforcement learning architecture and training methodology that use these datasets to create goal-driven agents that act and speak in the LIGHT environment; and (3) Empirical zero-shot evaluations based on human quest demonstrations and an analysis of large-scale transformer-based pre-training trends in static vs. interactive settings, showing that I have trained agents that act consistently and speak naturally with respect to their motivations.

7.1 LIGHT-Quests and ATOMIC-LIGHT

This section first provides a brief overview of the LIGHT game environment, followed by specific descriptions of the LIGHT-Quests and ATOMIC-LIGHT datasets used in this

Setting	You are in the Dangerous Precipice. The dangerous precipice overlooks the valley below. The ground slopes down to the edge here. Dirt crumbles down to the edge of the cliff. There's a dragon crescent, a knight's armor, a golden dragon egg, and a knight's fighting gear here. A knight is here.You are carrying nothing.
Partner:	Knight.
Persona	I am a knight. I come from a lower-ranking noble family. I serve under the king, as my father did before me. In times of war, I fight on horseback.
Carrying	knight's armor, golden dragon egg, knight's fighting gear
Self:	A dragon.
Persona	I am a dragon living in the mountains. I enjoy hoarding treasure. I terrorize the local populace for fun.
Carrying	Nothing.

Figure 7.1: Setting and character information for both self and partner characters as taken from LIGHT.

Motivations:		**Timeline:**	
Short	I need to recover the dragon egg that was stolen and punish the knight.	-4 hours	go to dangerous precipice
		-15 min	get knights armor from knight
		-10 min	get golden dragon egg
Mid	I need to return the golden dragon egg to my treasure hoard.	Now	hit knight
		+5 min	put dragon egg on back
		+15 min	eat the knight
Long	I need to build the largest hoard ever attained by any one dragon.	+2 hours	go to the mountains

Figure 7.2: Motivations with different levels of abstractions and corresponding sequence of timeline actions in chronological order for the self character in LIGHT-Quests. There are 7486 quests in total.

Insssssolent pessst! I should immolate you for this tressssspasss.

And why is that, dragon?

Ssstealing my precciousss golden egg! I'll tell you what, I'll give you 10 sssseconds to amussse me with your sssstory and THEN I'll burn you alive!

You said you wanted to attack me, dragon, did you not?

Go ahead, I'm lisssssstening.
get golden dragon egg

Now now! I would have given you that had you asked!

Asssssssk for my own property back? What a riduculousss notion

Look here, I told you to watch your mouth and you didn't, so leave or I'll make you leave.

And now threatsss! Thisss is proving to be a mossst engaging conversssation.
hit knight
Give my regardsss to the valley floor below!

Figure 7.3: Example of a demonstration of a human (blue shaded) completing the above quest while role-playing as the self character with a partner agent (grey shaded). There are 2111 such human demonstrations of average sequence length 12.92, consisting of 22672 dialogues in total.

chapter.

As mentioned in Chapter 2, recall that LIGHT game environment is a multi-user fantasy text-adventure game consisting of a rich, diverse set of characters, locations, and objects (1775 characters, 663 locations, and 3462 objects). Characters are able to perform templated actions to interact with both objects and characters, and can speak to other characters through free form text. Actions in text games generally consist of verb phrases (VP)

followed optionally by prepositional phrases (VP PP). For example, *get OBJ, put OBJ, give OBJ to CHAR*, etc.. There are 13 types of allowed verbs in LIGHT. These actions change the state of the world which is expressed to the player in the form of text descriptions.

7.1.1 LIGHT-Quests

Figures 7.1, 7.2, and 7.3 summarize the data that I collected for LIGHT-Quests. Data is collected via crowdsourcing in two phases, first the quests then demonstration of humans playing them. During the first phase, crowdworkers were given a setting, i.e. situated in a world, in addition to a character and its corresponding persona and asked to describe in free form text what potential motivations or goals could be for that character in the given world. The kind of information given to the crowdworkers is seen in Figure 7.1. Simultaneously, they were also asked to provide a sequence of seven timeline actions—one action that needs to be completed *now* and three before and after at various user-defined intervals—for how the character might go about achieving these motivations.

Given the information in Figure 7.1, the crowdworkers completed the above outlined tasks and produce data as seen in Figure 7.2. Motivations come in three levels of abstraction—short, mid, and long—corresponding to differing amounts of the timeline. For example, the short motivation is always guaranteed to correspond most closely to the *now* position on the timeline. Action annotation is pre-constrained based on the classes of verbs available within LIGHT. The rest of the action is completed as free form text as it may contain novel entities introduced in the motivations. There are 5982 training, 756 validation, and 748 test quests. Further details regarding the exact data collection process and details of LIGHT-Quests are found in Appendix B.4.

After collecting motivation and timelines for the quests, I deployed a two-player version of the LIGHT game, letting players attempt the quests for themselves in order to collect human demonstrations. Figure 7.3 shows an example human expert demonstration of a quest. Players were given a character, setting, motivation, and a partner agent and left to freely act

in the world and talk to the partner in pursuit of their motivations. The partner agent is a fixed poly-encoder transformer model (Humeau *et al.* 2020) trained on the original LIGHT data as well as other human interactions derived via the deployed game—using 111k utterances in total. Players first receive a role-playing score on a scale of 1-5 through a Dungeon Master (DM), a learned model that ranks how likely their utterances are given the current context. Once they have accumulated a score reaching a certain threshold, they are allowed to perform actions. I employ this gamification mechanism to encourage players to role-play their character persona and its motivations, leading to improved user experience and data quality (Horsfall and Oikonomou 2011). They are then given further reward if the actions they perform sequentially match those on the timeline for the given quest. The game ends after a maximum of six turns of dialogue per agent, i.e. twelve in total. The average sequence of a human demonstration is 12.92, with an average action sequence length of 2.18 and dialogue of 10.74. There are 1800 training, 100 validation, and 211 test human expert demonstrations after the data was filtered. Additional details and examples are found in Appendix B.4.1.

7.1.2 ATOMIC-LIGHT

Commonsense reasoning is a critical cornerstone when building learning agents that navigate spaces such as LIGHT-Quests. To this end, I domain-adapt the large-scale commonsense knowledge base ATOMIC (Sap *et al.* 2019) to LIGHT. ATOMIC contains information relevant for everyday commonsense reasoning in the form of typed if-then relations with variables. ATOMIC is organized into a set of events, e.g. "X puts X's trust in Y" and annotated relation types such as "needs", "wants", "attributes", and "effects" that label the effects. It is designed to be a general atlas of commonsense data and so is neither dependent on a specific environment or a character's persona and motivations.

To construct ATOMIC-LIGHT, I specifically use the relations for "intents", "effects", "wants" and "needs" and expand the $\langle subject, relation, object \rangle$ triples found in the graph

into templated natural language sentences. These sentences are then rewritten to better reflect the fantasy LIGHT domain. Named entities and other noun phrases in ATOMIC are masked out and filled in using BERT (Devlin *et al.* 2019) fine-tuned using a masked language model loss on the entire LIGHT and LIGHT-Quests data. I investigate the benefits of such domain adaptation on downstream tasks in Section 7.2.3. An example of a clause using the *wants* relation in ATOMIC is as follows, "*PersonX puts PersonX trust in PersonY*, *wants*, *rely on PersonY*." In ATOMIC-LIGHT, this is rewritten to: "The merchant puts the merchant's trust in the guard, as a result the merchant *wants* to rely on the guard." Similarly, an example of an effect using the *needs* relation is, "Before, the merchant puts the merchant's trust in the guard, the merchant *needs* to be friends with the guard." ATOMIC-LIGHT contains 216686 training, 35340 validation, and 38565 test samples.

Here I present 3 examples from ATOMIC-LIGHT for each of the 4 relation types used: "wants", "needs", "intents", and "effects".

```
[Effect] princess explains briefly the situation , as a result, princess points finger
[Effect] goblin king's healer provides care for patients , as a result, goblin king's
    healer assists patients
[Effect] witch changes men's appearance , as a result, witch causes men stress
[Want] prince plays a commander in the war, as a result, prince wants to win
[Want] repentant person focuses purely on issues, as a result, repentant person wants to
    help others
[Want] undead warrior hardens pharaoh's mind, as a result, undead warrior wants to make
    pharaoh punish people
[Intent] bandit plays a hand in the war because bandit wanted to participate
[Intent] ambassador focuses only on issues because ambassador wanted events to play out a
    certain way
[Intent] son proposes another plan because son wanted to be helpful
[Need] shipwrecked survivor proposes another wayward plan because shipwrecked survivor
    needed to leave this place
[Need] general proposes another way because general needed to come up with a proposal
[Need] citizen kills animals for food because citizen needed to learn to hunt
```

7.2 Agents that Act and Speak

This section describes the creation of the agents that learn to act and speak conditioned on their motivations in the LIGHT environment. The overall architecture and training are first outlined, followed by a detailed discussion on types of encoder pre-training.

Formally, I adapt the definition of text-based games as seen in Hausknecht *et al.* 2020; Côté *et al.* 2018 to LIGHT. They are partially observable Markov decision processes (POMDPs), represented as a 7-tuple of $\langle S, T, \mathcal{A}, \Omega, O, \mathcal{R}, \gamma \rangle$ representing the set of environment states, conditional transition probabilities between states, the vocabulary or words used to compose action commands or dialogue utterances (e.g. *get sword* or *Hey, give me that sword!* respectively), observations returned by the game, observation conditional probabilities, reward function, and the discount factor respectively. The LIGHT environment further allows us to factorize the overall action space $\mathcal{A}$ into A as the set of possible textual actions or commands (e.g. *get sword, steal coins from merchant*), and U as the set of possible dialogues that can be uttered by an agent, thus making it a factored POMDP (Degris and Sigaud 2013). This in turn means that, for a given quest q, each expert human demonstration $\mathcal{D}(q) = \alpha_0^*, \alpha_1^* ... \alpha_n^*$ can be factorized into two sub-sequences of expert demonstrations of actions and dialogue $\mathcal{D}_A(q) = a_0^*, a_1^*, ... a_n^*$ and $\mathcal{D}_U(q) = u_0^*, u_1^*, ... u_m^*$ respectively. The factorized action spaces A and U are constructed by enumerating all possible actions/dialogue utterances in the all human demonstrations in LIGHT-quests—$A = \bigcup_{q \in Q} \mathcal{D}_A(q); U = \bigcup_{q \in Q} \mathcal{D}_U(q)$ with $|A| = 4710$ and $|U| = 22672$.

7.2.1 LIGHT RL Environment

The environment as seen in Figure 7.4 consists of three components. The first is a partner agent, which is a model trained to play other agents in the game, as in Prabhumoye *et al.* 2020. Next is the game engine, which determines the effects of actions on the underlying game graph (Urbanek *et al.* 2019). Finally, there is the Dungeon Master (DM), which is

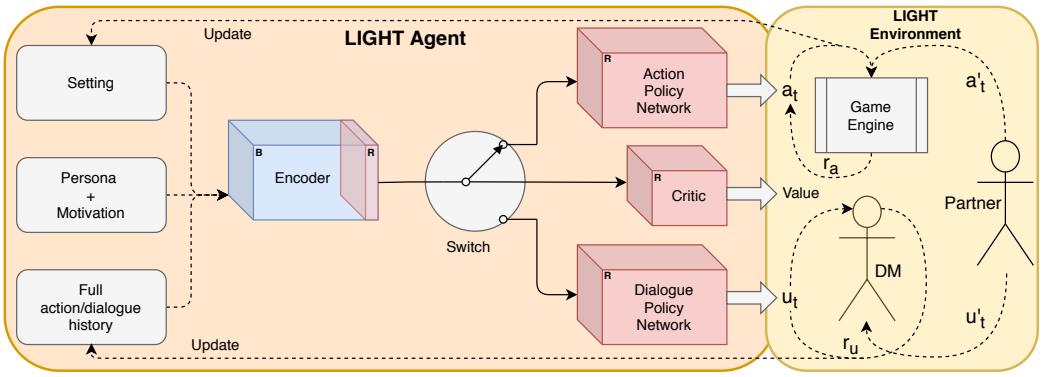


Figure 7.4: Overall RL Switch architecture and process. **B**lue shaded components can be pre-trained and **R**ed shaded components are trained with RL. Solid lines indicate gradient flow.

trained to score the naturalness of dialogue.

Partner Agent. The partner agent is a poly-encoder transformer model (Humeau *et al.* 2020) that is pre-trained on the Reddit dialogue corpus, then on LIGHT and the human demonstrations of LIGHT-Quests. Following the format seen in Figure 7.3, the partner agent does not have a motivation itself but is trained to react to agents with motivations. Following Prabhumoye *et al.* 2020, I keep the partner model fixed during the episodes where the LIGHT agent trains to ensure that it retains natural English semantics—avoiding the problem of language drift by learning an emergent language with that must agree with the partner's usage (Lee *et al.* 2019).

Action Rewards via the Game Engine. All actions, either those of the agent-in-training or the partner agent, are processed by the engine, checking for goal state completion—hence known as *act goals*. For example, if the LIGHT agent had the motivation to acquire a sword, the goal could be completed via a:

1. **self act completion**: where the agent acquires a sword itself by picking it up, stealing it, convincing the partner to drop theirs so you can pick it up, etc.

2. **partner act completion**: where the agent uses speech to convince their partner to achieve the goal for them (e.g., by persuading the partner to give them the sword).

Reaching an *act goal* provides reward r_a of 1 and 0 otherwise. At each step, the engine also

provides us with the set of valid actions. These are the subset of the action space A which are guaranteed to be a valid change to the world from the current state s_t, i.e. an action to give your partner a sword cannot be valid unless you possess the sword.

Speech Rewards via the Dungeon Master. Following prior works on using transformers for automatic evaluation of natural language generation (Sellam *et al.* 2020), I utilize a learned model–the Dungeon Master (DM)—to score the agent's ability to speak. The DM used here is a poly-encoder model trained on collected human quest demonstrations as well as the original conversations in LIGHT. It is conditioned on quests and motivations and thus able to provide a (noisy) indication of how natural the agent's dialogue utterances are given its immediate context, similarly to the function of the DM during the data collection process. Given the dialogue portion of a human quest demonstration of length n, the DM returns a reward r_u of $\frac{1}{2n}$ if an utterance was in the demonstration (for a maximum of one time per episode for each utterance from the demonstration). A further $\frac{1}{2n}$ is given each time the utterance is scored as being within the top-k most likely utterances by the DM. This naturalness objective will be hence referred to as a *speech goal*. These rewards thus also denser than *act goals*, helping the agent learn overall. Further, similarly to the game engine, the DM also provides a set of M valid utterances which are the M most likely dialogue candidates from the candidate set for the current context.

7.2.2 Training a LIGHT agent with Switch Reinforcement Learning

The overall architecture of my agent is shown in Figure 7.4. It consists of an encoder, a switch, an action network, and a dialogue network. First, I construct the action spaces—factorized into actions and utterances. The possible actions are the set of all actions taken in the demonstrations (4710 total) and the possible utterances are all utterances from the demonstrations (22672 total). The encoder network processes the setting, persona, motivation, as well as the full history of actions and dialogues performed by the agent and the partner, input as a text sequence. The features from the encoder, which here are the hidden

states at the final layer of a transformer, are used as input by all following components of the agent. In Section 7.3 I show how different encoder training data affects the model.

Next, a switch module makes the decision regarding whether the agent should act or talk in the current context and activates the corresponding policy network. In this work, the switch is simple: it outputs an action every k dialogue utterances; where during training k is chosen to match the ratio of utterances to actions on that particular quest from the human demonstrations, and during testing, k is chosen to match the average action to utterance ratio. Both the action and dialogue policies consist of a a single GRU layer followed by an n-layer feed-forward network given input features from the encoder. Once the LIGHT agent has output an utterance or action, it is processed by the environment—the partner agent, the game engine and the DM.

I use A2C (Mnih *et al.* 2016) to train the LIGHT agent, treating the two policy networks as two separate actors with a shared critic. The shared critic is motivated by the concepts of *self act completion* and *partner act completion* seen in Section 7.2.1 where the LIGHT agent can speak to convince the partner to achieve an *act goal*. Each agent in a batch is initialized via priority sampling (Graves *et al.* 2017) with a different quest, i.e. quests that the agent has historically successfully completed less often are given a greater weight when sampling from the pool of all possible training quests. In addition to a normal entropy regularization term, I also add a regularization term that encourages the models to produce "valid" outputs as judged by the game engine and the DM for actions and utterances respectively.

Following Chapter 4, I introduce two additional entropy loss terms to speed up exploration.

$$\mathcal{L}_{\mathbb{A}}(s_t, a_t; \theta_{A_t}) = \sum_{i=1}^{N} (y_{a_i} log \pi_{\mathbb{A}}(a_i|s_t) + (1 - y_{a_i})(1 - log \pi_{\mathbb{A}}(a_i|s_t)) \tag{7.1}$$

$$\mathcal{L}_{\mathbb{U}}(s_t, u_t; \theta_{U_t}) = \sum_{i=1}^{M} (y_{u_i} log \pi_{\mathbb{U}}(u_i|s_t) + (1 - y_{u_i})(1 - log \pi_{\mathbb{U}}(u_i|s_t)) \tag{7.2}$$

$$y_{a_i} = \begin{cases} 1 & a_i \in A_{valid}(s_t) \\ 0 & else \end{cases} \quad y_{o_i} = \begin{cases} 1 & u_i \in U_{valid}(s_t) \\ 0 & else \end{cases}$$

Each of these loss terms are only applied to the relevant policy network, i.e. $\mathcal{L}_{\mathbb{A}}$ to the action network and $\mathcal{L}_{\mathbb{U}}$ to the dialogue network. These terms provide an additional training signal to the policy networks regarding which actions and dialogue are contextually relevant via additional entropy regularization over the valid actions. Similarly to the results found in Chapter 4,, preliminary experiments in this domain suggest that these terms reduce the number of environment steps required to reach asymptotic performance by a couple orders of magnitude.

A2C is a policy gradient algorithm that maximizes long-term expected reward by comparing the advantage $A(s_t, a_t^*)$ of taking an action in a state to the average value of taking a valid action as predicted by the critic $V(s_t)$.

$$A(s_t, a_t^*) = \mathbb{E}[r_t + \gamma V(s_{t+1})] - V(s_t) \text{ where } r_t = r_{A_t} + r_{U_t} \tag{7.3}$$

Here, a_t^* is either an action or an utterance outputted by the respective policy networks. It is also worth noting that on steps where an action is performed, r_{U_t} is always 0, but on steps where a dialogue utterance is spoken r_{A_t} may not be 0. This corresponds to the concepts of *self act completion* and *partner act completion* seen in Section 7.2.1 where the LIGHT agent can speak to convince the partner to achieve an *act goal*. This motivates having a shared critic between the policy networks. Both policies are then updated according

to the gradient

$$-\nabla_\theta \begin{cases} log\pi_A(a_t|s_t;\theta_{A_t})A(s_t,a_t) + \mathcal{L}_{\mathbb{A}}(s_t,a_t;\theta_{A_t}) + \sum_{a\in A} P(a|s_t)logP(a|s_t) \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \pi_{\mathbb{S}}(s_t) = \pi_A \\ log\pi_U(u_t|s_t;\theta_{U_t})A(s_t,u_t) + \mathcal{L}_{\mathbb{U}}(s_t,u_t;\theta_{U_t}) + \sum_{u\in U} P(u|s_t)logP(u|s_t) \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \pi_{\mathbb{S}}(s_t) = \pi_U \end{cases} \tag{7.4}$$

The additional terms seen are an overall entropy loss over the entire action A or utterance U spaces, designed to prevent premature, sub-optimal policy convergence. Boltzmann exploration (Sutton and Barto 1998) is used to sample actions from both actor networks during training.

7.2.3 Encoder Pre-training Tasks

Prior work on commonsense reasoning in supervised natural language learning (Bosselut *et al.* 2019) suggests that the encoder is key to overcoming the challenges posed by the LIGHT-Quests dataset even in an RL setting. I describe a series of encoder pre-training tasks, designed to help the LIGHT agent either act more consistently or speak more naturally.

ATOMIC-LIGHT. As seen in Section 7.1, ATOMIC-LIGHT is a (domain-adapted) fantasy commonsense knowledge graph, and as such provides priors for an agent on how to act consistently in the world. For example, given a clause such as "The knight wishes to slay the dragon, as a result the knight *needs* to acquire a sword," the task would be to predict the underlined text—a form of knowledge graph completion (Wang *et al.* 2017).

Reddit. I use a previously existing Reddit dataset extracted and obtained by a third party and made available on pushshift.io (Baumgartner *et al.* 2020) seen in Roller *et al.* 2020. This dataset has been used in several existing dialogue-based studies and has been shown to result in more natural conversations (Yang *et al.* 2018; Mazaré *et al.* 2018).

LIGHT-Original. The original LIGHT dataset (Urbanek *et al.* 2019) is organized sim-

ilarly to the human demonstrations found in LIGHT-Quests, i.e. an interspersed sequence of dialogue and actions collected from humans role-playing a character. The task itself is to predict the next action or utterance given the prior dialogue history as well as the current setting and persona for a character. They are collected in a chit-chat fashion, with no notion of objectives, and so provide priors on how to generally act consistently and speak in a fantasy world, but not directly how to complete quests.

LIGHT-Quests. Pre-training with this newly introduced dataset consists of three tasks. (1) *Bag-of-action timeline prediction* in which, given a quest consisting of setting, persona, and motivations, any one of the actions in the timeline must be predicted. (2) *Sequential timeline prediction* in which, given a quest consisting of setting, persona, motivations, and the first n actions in the timeline, the $n + 1^{th}$ action must be predicted. (3) Predict the next dialogue utterance given a human demonstration in a manner similar to the LIGHT-original tasks. The first two tasks are designed to help the agent act consistently and the third to help it speak naturally with respect to its motivations.

7.3 Evaluation

I conduct two ablation studies, (1) to compare the effects of the encoder pre-training tasks in RL settings vs. supervised behavior cloning, and (2) to analyze the interplay between actions and dialogue for *self* and *partner act completions*.

7.3.1 Encoder Pre-training Type Ablation Study

Pre-training is done on the tasks described in Section 7.2.3 by training a 12 layer transformer with 256 million parameters using a cross-entropy loss as seen in (Humeau *et al.* 2020). As seen in Figure 7.4, weights are then transferred to the **B**lue shaded portion of the encoder as seen in Figure 7.4 and frozen. A further three randomly initialized-layers are appended on to the end, indicated by the **R**ed portions, into which gradients flow. This is done as optimizing all the parameters of such a model via RL over a long horizon is

both data inefficient and computationally infeasible. Additional hyperparameter details are found in Appendix 7.3.1. I investigate the following five different pre-training models to see how they compare on *act* and *speech goal completions* when trained with RL and in a supervised manner with behavior cloning:

Scratch. No pre-training is done, the encoder is a 3-layer randomly initialized transformer and trained along with the policy networks.

General. Multi-task trained using both pushshift.io Reddit and the commonsense dataset ATOMIC-LIGHT, giving the agent general priors on how to act and speak.

Light. Multi-task trained on all tasks in LIGHT-original and LIGHT-Quests, giving the agent priors on how to act and speak with motivations in the LIGHT fantasy domain.

General+Light. Multi-task trained on all tasks used in the General and Light models.

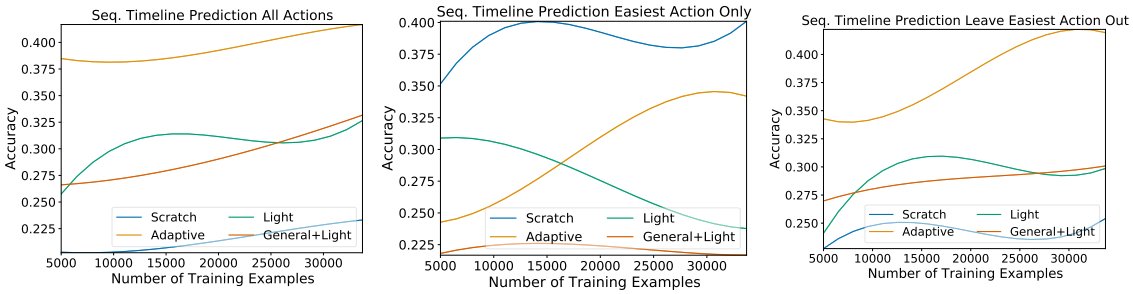
Adaptive. Here I adaptively train a General+Light model that is first initialized itself from a General model, providing additional regularization to help balance between Light and General tasks.

Table 7.1: Sequential supervised timeline prediction.

Model	**All Actions**			**Easiest Action**	**Leave Easiest Out**
	Hits@1	**Hits@5**	**Hits@10**	**Hits@1**	**Hits@1**
Scratch	0.2332	0.7491	0.9176	**0.4013**	0.2546
No Motivations	0.1132	0.5412	0.5771	0.1886	0.164
Short Motivations	0.1856	0.6479	0.678	0.261	0.223
Long & Mid Motivations	0.1452	0.598	0.631	0.2241	0.1272
Light	0.3156	0.7854	0.9226	0.236	0.2968
General+Light	0.311	0.7772	0.9229	0.2173	0.2995
Untuned ATOMIC	0.274	0.761	0.909	0.1912	0.2677
Adaptive	**0.4168**	**0.8012**	**0.9332**	0.342	**0.4194**
No Motivations	0.16	0.6286	0.6415	0.2838	0.1966
Short Motivations	0.225	0.6592	0.8245	0.305	0.2106
Long & Mid Motivations	0.1682	0.6397	0.6499	0.281	0.1595

Supervised Pre-training Results

This section describes results from the LIGHT-Quests tasks that are described in Section 7.2.3. Model-types are the same as those used in the encoders in Section 7.3. All

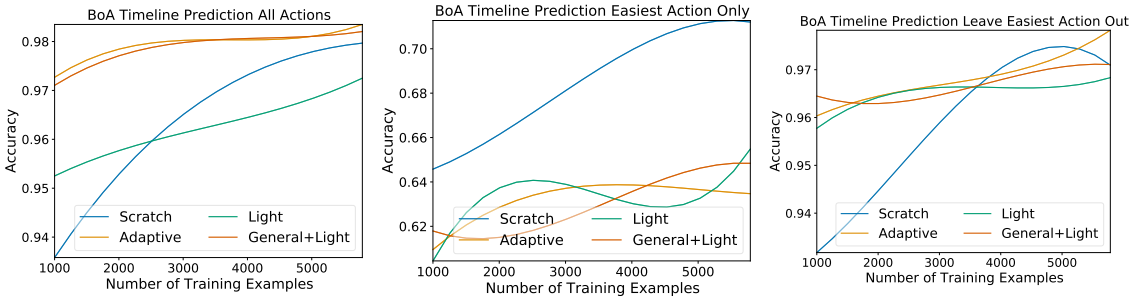


(a) All Actions (b) Easiest Action Only (c) Leave Easiest Action Out

Figure 7.5: Sequential supervised timeline prediction learning curves.

Table 7.2: Bag of Actions supervised timeline prediction.

Model	All Actions Hits@1	All Actions Hits@5	All Actions Hits@10	Easiest Action Hits@1	Leave Easiest Out Hits@1
Scratch	0.9791	1	1	**0.7122**	0.9721
No Motivations	0.901	1	1	0.554	0.8823
Short Motivations	0.934	1	1	0.622	0.9211
Long & Mid Motivations	0.921	1	1	0.5679	0.956
Light	0.9721	1	1	0.6552	0.9682
General+Light	0.9818	1	1	0.6472	0.9708
Untuned ATOMIC	0.9421	1	1	0.6272	0.9508
Adaptive	**0.9829**	1	1	0.6353	**0.9768**
No Motivations	0.9175	1	1	0.5756	0.9523
Short Motivations	0.9794	1	1	0.6578	0.9682
Long & Mid Motivations	0.9523	1	1	0.5812	0.9576



(a) All Actions (b) Easiest Action Only (c) Leave Easiest Action Out

Figure 7.6: Bag of Actions supervised timeline prediction learning curves.

retrieval results reported are Hits@X/100. Results are reported for all timeline actions, all actions with the exception of the easiest action—the action at the "now" position in the timeline, corresponding most closely to the short motivation as a result of the framing of Mechanical Turk task—and only the easiest action prediction.

Tables 7.1, 7.2 and Figures 7.5, 7.6 summarize these results. Some notable common trends across these tasks are:

1. Removing motivations from the input context results in significantly lower performance—on average ≈ 7 points lower accuracy for Bag of Actions Timeline prediction and on average ≈ 18 percentage points lower for Sequential Timeline prediction when averaged across Scratch and Adaptive models. Further, the short motivations proves to be the most useful for timeline prediction tasks.

2. Pre-training on ATOMIC-LIGHT produces an average *gain* of ≈ 4 percentage points in accuracy in both tasks than when trained on ATOMIC without domain adaptation alone.

3. Performance across the board increases with an increase in the number of training quests, as seen in Figures 7.5, 7.6, with the Scratch model receiving the greatest benefit from having more training data.

4. The Scratch model performs "best" on evaluations for the easiest action only but no others—indicating that it has overfit to predicting the easiest action which closely corresponds to short motivation. Likewise, the Adaptive generally has the lowest performance for the easiest action—indicating that pre-training with the other tasks has provided sufficient regularization to enable it to not overfit to the easiest action.

Reinforcement Learning

Table 7.3 describes the reinforcement learning results for this ablation. Models were each zero-shot evaluated on 211 human demonstrations from the LIGHT-Quests test set for a single episode per quest across three independent runs. Figure 7.7 shows learning curves during training for each encoder type. I first see that performance when trained with RL, i.e. with interactivity and environment grounding during training, results in higher performance

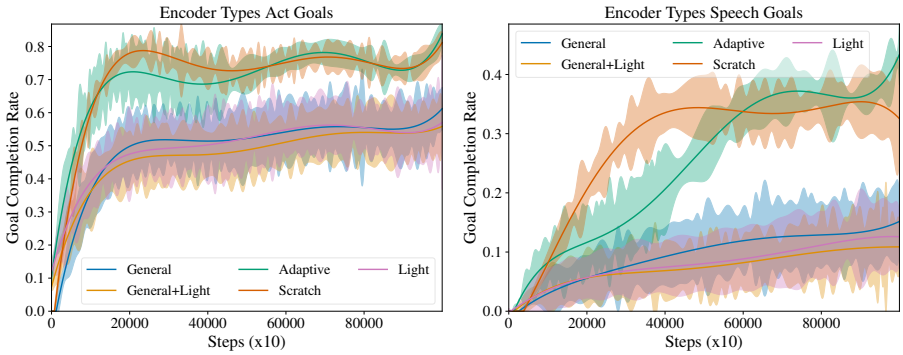


Figure 7.7: Encoder types RL reward curves averaged over 3 independent runs.

Table 7.3: Encoder Type RL Zero-Shot Evaluations averaged over 3 independent runs. Act goals and speech goals are as described in Section 7.2.1. Standard deviations for all experiments are less than 0.01. The "Act & Speech Goals" column refers to quests where the agent has simultaneously achieved both types of goals within the episode. Human act goal completion $= 0.6$ as measured during the second phase of the LIGHT-Quests data collection.

Model	**Reinforcement Learning**			**Behavioral Cloning**
	Act Goals	**Speech Goals**	**Act & Speech Goals**	**Act & Speech Goals**
Scratch	0.418	0.118	0.103	0.0003
General	0.146	0.040	0.028	0.00226
Light	0.115	0.028	0.022	0.0934
General+Light	0.251	0.094	0.081	0.115
Adaptive	**0.420**	**0.330**	**0.303**	**0.147**

than behavioral cloning for all the models. In both RL and behavior cloning settings the Adaptive model outperforms all others in all the metrics.

When trained supervised (behavioral cloning), I see trends mirroring standard pre-training in static text corpora. Transfer is easy and the Scratch model performs significantly worse than all others; and each new task added improves the agent's ability to speak and act. In particular, I see that Light outperforms General, showing that the more similar the pre-training tasks are to the downstream tasks, the better the supervised performance.

However, these trends do not hold in the RL setting. The Scratch model outperforms everything except the Adaptive model and General outperforms Light. In part, this may be due to specification gaming (Krakovna *et al.* 2020); however Adaptive does strongly outperform Scratch in goals with dialogue. This suggests that transfer (and fine-tuning) is not as simple in the RL setting as in the supervised setting, but still can be useful if carefully

Table 7.4: Ability type ablations averaged across 3 runs with standard deviations less than 0.01.

Ability	Scratch			Adaptive		
	Act	**Speech**	**Act & Speech**	**Act**	**Speech**	**Act & Speech**
Act+Speech	0.418	0.118	**0.103**	0.420	0.330	**0.303**
Act Only	**0.478**	-	-	**0.469**	-	-
Speech Only	0.036	**0.165**	0.028	0.0398	**0.341**	0.030
-No Speech Goals	0.0526	0.0521	0.0331	0.0673	0.0947	0.041

done. I note that domain adapative pre-training (intermediate task transfer) has previously been shown to give modest gains in supervised learning (Phang *et al.* 2018; Gururangan *et al.* 2020), but not with the large effects seen here for RL. Figure 7.7 further shows that with the right combination of tasks, not only is the generalization performance better, but training itself is more sample efficient—requiring fewer steps before reaching asymptotic performance.

When comparing Scratch and Adaptive in the RL setting, I see that the Scratch's *act goal completion* rates are only marginally lower than the Adaptive, but the Adaptive has much higher *speech goal completion* resulting in an overall higher joint *act & speech goal completion* rates. Additionally, in Figure 7.7 I see that during training, both Scratch and Adaptive reach similar goal completion rates for act and speech but the gap in zero-shot performance on novel data is much greater for speech than act. I hypothesize that this, in-part, is due to the differences in reward functions. *Act goals* are decided by the game engine and constitute ground truth, whereas *speech goals* during training are at least partially decided by the DM, a learned and therefore more noisy indicator of the ground truth. Speech is thus the harder sub-task in my formulation and therefore receives more benefits from pre-training.

7.3.2 Ability Type Ablation Study

To better understand the interplay between acts and speech resulting in *self* and *partner act goal completions*, I perform an ablation study selectively dropping either the agent's ability to talk or act. I train the agent to either only act, only speak, only speak with only action

rewards. In the scenarios when the agent can only speak, the agent has to convince the partner to help achieve the agent's goal.

The results are outlined in Table 7.4. Unsurprisingly, when trained to only act, the act goal completion rate increases over when it can both act and speak. Similarly, when trained to only speak the speech goal completion rates also increase. I can draw two conclusions from these results: (1) It is much easier to do an action yourself than to convince the partner to do it (2) Removing speech goals increases the act goal completion rates corresponding to higher partner act completions. Thus, the sequences of dialogue utterances required to convince the partner to achieve the agent's goal are likely often at odds with those sequences required to maximize speech goals.

7.4 Conclusions

Operating on the hypothesis that interactivity is key to language learning, I introduce two datasets—a set of quests based on character motivations in fantasy worlds, LIGHT-Quests, and a large-scale commonsense knowledge graph, ATOMIC-LIGHT—and a reinforcement learning system that leverages transformer-based pre-training to facilitate development of goal-driven agents that can act and speak in situated environments. Zero-shot evaluations on a set of novel human demonstration show that I have trained agents that act consistently and speak naturally with respect to their motivations. A key insight from my ablation study testing for zero-shot generalization on novel quests is that large-scale pre-training in interactive settings require careful selection of pre-training tasks—balancing between giving the agent "general" open domain priors and those more "specific" to the downstream task—whereas static methodologies require only domain specific pre-training for effective transfer but are ultimately less effective than interactive methods.

CHAPTER 8
WORLD GENERATION

A core component of many narrative-based tasks—everything from storytelling to game generation—is world building. The world of a story or game defines the boundaries of where the narrative is allowed and what the player is allowed to do. There are four core challenges to world generation: (1) commonsense knowledge: the world must reference priors that the player possesses so that players can make sense of the world and build expectations on how to interact with it. This is especially true in interactive fictions where the world is presented textually because many details of the world necessarily be left out (e.g., the pot is on a stove; kitchens are found in houses) that might otherwise be literal in a graphical virtual world. (2) Thematic knowledge: interactive fictions usually involve a theme or genre that comes with its own expectations. For example, light speed travel is plausible in sci-fi worlds but not realistic in the real world. (3) Coherence: the world must not appear to be an random assortment of locations. (3) Natural language: The descriptions of the rooms as well as the permissible actions must be text, implying that the system has natural language generation capability.

Because worlds are conveyed entirely through natural language, the potential output space for possible generated worlds is combinatorially large. To constrain this space and to make it possible to evaluate generated world, I present an approach which makes use of existing stories, building on the worlds presented in them but leaving enough room for the worlds to be unique. Specifically, I take a story such as Sherlock Holmes or Rapunzel—a linear reading experience—and extract the description of the world the story is set in to make an interactive, explorable world.

My method first extracts a partial, potentially disconnected knowledge graph from the story, encoding information regarding locations, characters, and objects in the form of

Bank vault
It is about three feet in height, and one and a half in width.
Exits: Baker Street and Wilson's shop
You see: Archie, Helper and John Clay

Action: **Examine John Clay**

John Clay
Short, stocky, and one of the taller kind, John Clay is the kind of man who lives and dies by the watch he keeps.

Action: **Go to Baker Street**

Baker Street
Like any other street in London, it is a-stage-set, with the best and the worst of society crammed into one place.

Figure 8.1: Example player interaction in the deep neural generated mystery setting.

$\langle entity, relation, entity \rangle$ triples. Relations between these types of entities as well as their properties are captured in this knowledge graph. However, stories often do not explicitly contain all the information required to fully fill out such a graph. A story may mention that there is a sword stuck in a stone but not what you can do with the sword or where it is in relation to everything else. My method fills in missing relation and affordance information using thematic knowledge gained from training on stories in a similar genre. This knowledge graph is then used to guide the text description generation process for the various locations, characters, and objects. The game is then assembled using the knowledge graph and the corresponding generated descriptions.

I have two major contributions. (1) A neural model and a rules-based baseline for each of the tasks described above. The phases are that of graph extraction and completion followed by description generation and game formulation. Each of these phases are relatively distinct and utilize their own models. (2) A human subject study for comparing the neural model and variations on it to the rules-based and human-made approaches. I perform two separate human subject studies—one for the first phase of knowledge graph construc-

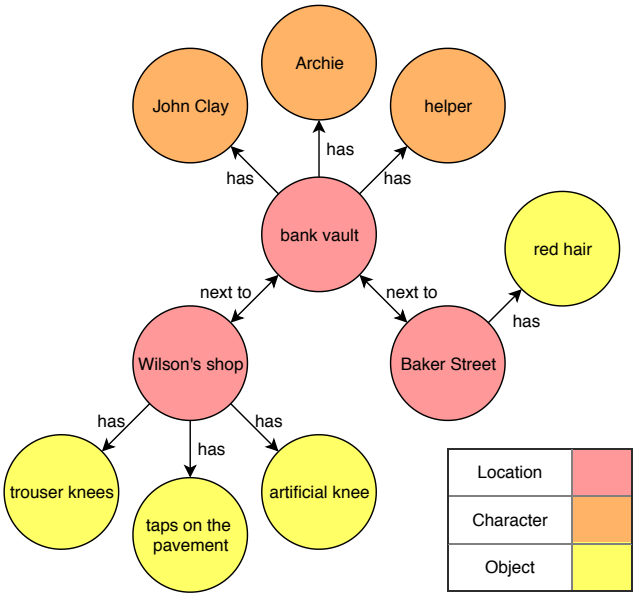


Figure 8.2: Example knowledge graph constructed by AskBERT.

tion and another for the overall game creation process—testing specifically for coherence, interestingness, and the ability to maintain a theme or genre.

8.1 Bringing Stories Alive

World generation happens in two phases. In the first phase, a partial knowledge graph is extracted from a story plot and then filled in using thematic commonsense knowledge. In the second phase, the graph is used as the skeleton to generate a full interactive fiction game—generating textual descriptions or "flavortext" for rooms and embedded objects. I present a novel neural approach in addition to a rule guided baseline for each of these phases in this section.

8.1.1 Knowledge Graph Construction

The first phase is to extract a knowledge graph from the story that depicts locations, characters, objects, and the relations between these entities. I present two techniques. The first uses neural question-answering technique to extract relations from a story text. The sec-

ond, provided as a baseline, uses OpenIE5,[1] a commonly used rule-based information extraction technique. For the sake of simplicity, I considered primarily the location-location and location-character/object relations, represented by the "next to" and "has" edges respectively in Figure 8.2.

Neural Graph Construction

While many neural models already exist that perform similar tasks such as named entity extraction and part of speech tagging, they often come at the cost of large amounts of specialized labeled data suited for that task. I instead propose a new method that leverages models trained for context-grounded question-answering tasks to do entity extraction with no task dependent data or fine-tuning necessary. My method, dubbed *AskBERT*, leverages the Question-Answering (QA) model ALBERT (Lan *et al.* 2019). AskBERT consists of two steps seen in Figure 8.3: vertex extraction and graph construction.

The first step is to extract the set of entities—graph vertices—from the story. I am looking to extract information specifically regarding characters, locations, and objects. This is done by using asking the QA model questions such as "Who is a character in the story?". Ribeiro *et al.* (2019) have shown that the phrasing of questions given to a QA model is important and this forms the basis of how I formulate my questions—questions are asked so that they are more likely to return a single answer, e.g. asking "Where is a location in the story?" as opposed to "Where are the locations in the story?". In particular, I notice that pronoun choice can be crucial; "Where is a location in the story?" yielded more consistent extraction than "What is a location in the story?". ALBERT QA is trained to also output a special $<no\text{-}answer>$ token when it cannot find an answer to the question within the story. My method makes use of this by iteratively asking QA model a question and masking out the most likely answer outputted on the previous step. This process continues until the $<no\text{-}answer>$ token becomes the most likely answer.

[1] https://github.com/dair-iitd/OpenIE-standalone

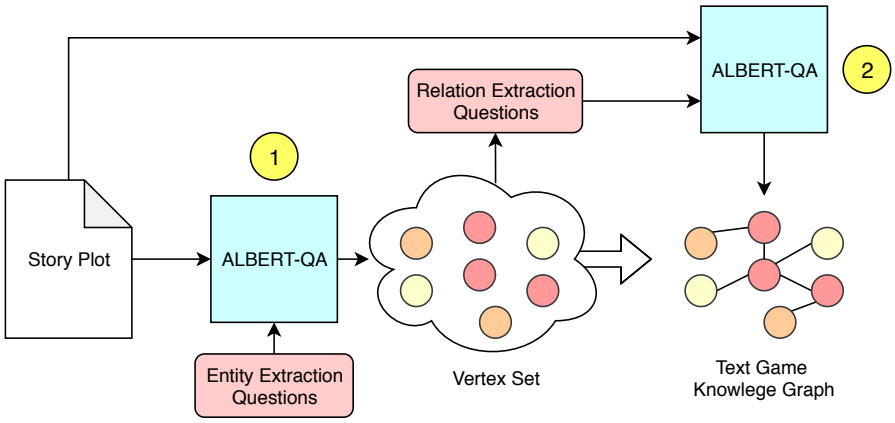


Figure 8.3: Overall AskBERT pipeline for graph construction.

The next step is graph construction. Typical interactive fiction worlds are usually structured as trees, i.e. no cycles except between locations. Using this fact, I use an approach that builds a graph from the vertex set by one relation—or edge—at a time. Once again using the entire story plot as context, I query the ALBERT-QA model picking a random starting location x from the set of vertices previously extracted.and asking the questions "What location can I visit from x?" and "Who/What is in x?". The methodology for phrasing these questions follows that described for the vertex extraction. The answer given by the QA model is matched to the vertex set by picking the vertex u that contains the best word-token overlap with the answer. Relations between vertices are added by computing a relation probability on the basis of the output probabilities of the answer given by the QA model. The probability that vertices x, u are related:

$$P(x,u) = \frac{p(x,u) + p(u,x)}{2} \tag{8.1}$$

where

$$p(x,u) = \sum_{o \in \text{QA outputs}} p(o) \mathbb{1}\{u = \operatorname*{argmax}_{v}(v \cap o)\} \tag{8.2}$$

is the sum of the individual token probabilities of the overlapping tokens in the answer from the QA model and u.

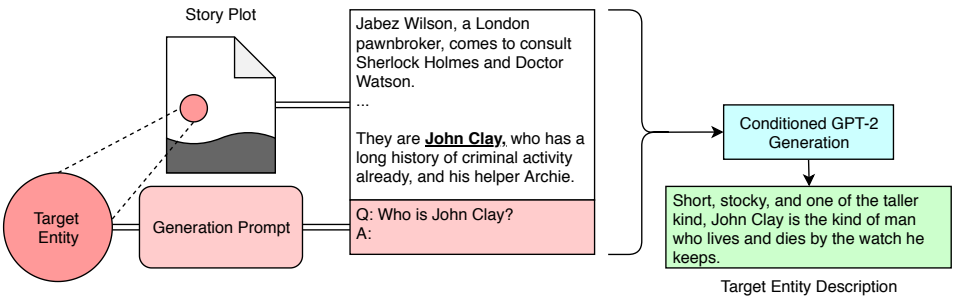


Figure 8.4: Overview for neural description generation.

Rule-Based Graph Construction

I compared my proposed AskBERT method with a non-neural, rule-based approach. This approach is based on the information extracted by OpenIE5, followed by some post-processing such as named-entity recognition and part-of-speech tagging. OpenIE5 combines several cutting-edge ideas from several existing papers (Saha and Mausam 2018; Pal and Mausam 2016; Christensen *et al.* 2011) to create a powerful information extraction tools. For a given sentence, OpenIE5 generates multiple triples in the format of $\langle entity, relation, entity \rangle$ as concise representations of the sentence, each with a confidence score. These triples are also occasionally annotated with location information indicating that a triple happened in a location.

As in the neural AskBERT model, I attempt to extract information regarding locations, characters, and objects. The entire story plot is passed into the OpenIE5 and I receive a set of triples. The location annotations on the triples are used to create a set of locations. I mark which sentences in the story contain these locations. POS tagging based on marking noun-phrases is then used in conjunction with NER to further filter the set of triples—identifying the set of characters and objects in the story.

The graph is constructed by linking the set of triples on the basis of the location they belong to. While some sentences contain very explicit location information for OpenIE5 to mark it out in the triples, most of them do not. I therefore make the assumption that

the location remains the same for all triples extracted in between sentences where locations are explicitly mentioned. For example, if there exists $locationA$ in the 1st sentence and $locationB$ in the 5th sentence of the story, all the events described in sentences 1-4 are considered to take place in $locationA$. The entities mentioned in these events are connected to $locationA$ in the graph.

8.1.2 Description Generation

The second phase involves using the constructed knowledge graph to generate textual descriptions of the entities I have extracted, also known as flavortext. This involves generating descriptions of what a player "sees" when they enter a location and short blurbs for each object and character. These descriptions need to not only be faithful to the information present in the knowledge graph and the overall story plot but to also contain flavor and be interesting for the player.

Neural Description Generation

Here, I approach the problem of description generation by taking inspiration from conditional transformer-based generation methods (Shirish Keskar *et al.* 2019). My approach is outlined in Figure 8.4 and an example description shown in Figure 8.1. For any given entity in the story, I first locate it in the story plot and then construct a prompt which consists of the entire story up to and including the sentence when the entity is first mentioned in the story followed by a question asking to describe that entity. With respect to prompts, I found that more direct methods such as question-answering were more consistent than open-ended sentence completion. For example, "Q: Who is the prince? A:" often produced descriptions that were more faithful to the information already present about the prince in the story than "You see the prince. He is/looks". For my transformer-based generation, I use a pre-trained 355M GPT-2 model (Radford *et al.* 2019) finetuned on a corpus of plot summaries collected from Wikipedia. The plots used for finetuning are tailored specific to

the genre of the story in order to provide more relevant generation for the target genre. Additional details regarding the datasets used are provided in Section 8.2. This method strikes a balance between knowledge graph verbalization techniques which often lack "flavor" and open ended generation which struggles to maintain semantic coherence.

Rules-Based Description Generation

In the rule-based approach, I utilized the templates from the built-in text game generator of TextWorld (Côté *et al.* 2018) to generate the description for my graphs. TextWorld is an open-source library that provides a way to generate text-game learning environments for training reinforcement learning agents using pre-built grammars.

Two major templates involved here are the Room Intro Templates and Container Description Templates from TextWorld, responsible for generating descriptions of locations and blurbs for objects/characters respectively. The location and object/character information are taken from the knowledge graph constructed previously.

- Example of Room Intro Templates: "This might come as a shock to you, but you've just $\#entered\#$ a $<location\text{-}name>$"
- Example of Container Description Templates: "The $<location\text{-}name>$ $\#contains\#$ $<object/person\text{-}name>$"

Each token surrounded by $\#$ sign can be expanded using a select set of terminal tokens. For instance, $\#entered\#$ could be filled with any of the following phrases here: entered; walked into; fallen into; moved into; stumbled into; come into. Additional prefixes, suffixes and adjectives were added to increase the relative variety of descriptions. Unlike the neural methods, the rule-based approach is not able to generate detailed and flavorful descriptions of the properties of the locations/objects/characters. By virtue of the templates, However, it is much better at maintaining consistency with the information contained in the knowledge graph.

8.2 World Generation Evaluation

I conducted two sets of human participant evaluations by recruiting participants over Amazon Mechanical Turk. The first evaluation tests the knowledge graph construction phase, in which I measure perceived coherence and genre or theme resemblance of graphs extracted by different models. The second study compares full games—including description generation and game assembly, which can't easily be isolated from graph construction—generated by different methods. This study looks at how interesting the games were to the players in addition to overall coherence and genre resemblance. Both studies are performed across two genres: mystery and fairy-tales. This is done in part to test the relative effectiveness of my approach across different genres with varying thematic commonsense knowledge. The dataset used was compiled via story summaries that were scraped from Wikipedia via a recursive crawling bot. The bot searched pages for both for plot sections as well as links to other potential stories. From the process, 695 fairy-tales and 536 mystery stories were compiled from two categories: novels and short stories. I note that the mysteries did not often contain many fantasy elements, i.e. they consisted of mysteries set in my world such as Sherlock Holmes, while the fairy-tales were much more removed from reality. Details regarding how each of the studies were conducted and the corresponding setup are presented below.

8.2.1 Knowledge Graph Construction Evaluation

I first select a subset of 10 stories randomly from each genre and then extract a knowledge graph using three different models. Each participant is presented with the three graphs extracted from a single story in each genre and then asked to rank them on the basis of how coherent they were and how well the graphs match the genre. The graphs resembles the one shown in in Figure 8.2 and are presented to the participant sequentially. The exact order of the graphs and genres was also randomized to mitigate any potential latent correlations.

Table 8.1: **Vertex statistics:** Average vertex count by type per genre. The random model has the same vertex statistics as the neural model.

Genre	Category	Neural	Rules
Mystery	Locations	7.2	3.5
	Characters	4.8	4.1
	Objects	3.2	12.2
Fairy-tale	Locations	4.0	1.8
	Characters	3.3	1.2
	Objects	4.1	8.7

Table 8.2: **Edge and degree statistics:** Average edge count , average degree count, and degree standard deviation of the graphs.

Genre	Statistic	Neural	Rules	Random
Mystery	Avg. Edges	10.7	22.3	10.7
	Avg. Degree	1.63 ± 1.77	2.15 ± 0.38	1.63 ± 1.63
Fairy-tale	Avg. Edges	16.7	12	16.7
	Avg. Degree	1.73 ± 2.04	1.98 ± 0.29	1.73 ± 1.64

Overall, this study had a total of 130 participants.This ensures that, on average, graphs from every story were seen by 13 participants.

In addition to the neural AskBERT and rules-based methods, I also test a variation of the neural model which I dub to be the "random" approach. The method of vertex extraction remains identical to the neural method, but I instead connect the vertices randomly instead of selecting the most confident according to the QA model. I initialize the graph with a starting location entity. Then, I randomly sample from the vertex set and connect it to a randomly sampled location in the graph until every vertex has been connected. This ablation in particular is designed to test the ability of my neural model to predict relations between entities. It lets us observe how accurately linking related vertices effects each of the metrics that I test for. For a fair comparison between the graphs produced by different approaches, I randomly removed some of the nodes and edges from the initial graphs so that the maximum number of locations per graph and the maximum number of objects/people per location in each story genre are the same.

The results are shown in Table 8.3. I show the median rank of each of the models for

both questions across the genres. Ranked data is generally closely interrelated and so I perform Friedman's test between the three models to validate that the results are statistically significant. This is presented as the p-value in table (asterisks indicate significance at $p < 0.05$). In cases where I make comparisons between specific pairs of models, when necessary, I additionally perform the Mann-Whitney U test to ensure that the rankings differed significantly.

In the mystery genre, the rules-based method was often ranked first in terms of genre resemblance, followed by the neural and random models. This particular result was not statistically significant However, likely indicating that all the models performed approximately equally in this category. The neural approach was deemed to be the most coherent followed by the rules and random. For the fairy-tales, the neural model ranked higher on both of the questions asked of the participants. In this genre, the random neural model also performed better than the rules based approach.

Tables 8.1 and 8.2 show the statistics of the constructed knowledge graphs in terms of vertices and edges. I see that the rules-based graph construction has a loIr number of locations, characters, and relations between entities but far more objects in general. The greater number of objects is likely due to the rules-based approach being unable to correctly identify locations and characters. The gap between the methods is less pronounced in the mystery genre as opposed to the fairy-tales, in fact the rules-based graphs have more relations than the neural ones. The random and neural models have the same number of entities in all categories by construction but random in general has loIr variance on the number of relations found. In this case as well, the variance is loIr for mystery as opposed to fairy-tales. When taken in the context of the results in Table 8.3, it appears to indicate that leveraging thematic commonsense in the form of AskBERT for graph construction directly results in graphs that are more coherent and maintain genre more easily. This is especially true in the case of the fairy-tales where the thematic and everyday commonsense diverge more than than in the case of the mysteries.

Table 8.3: Results of the knowledge graph evaluation study.

Genre	Questions	Neural	Rules	Random	p-value
Mystery	Resembles Genre	2	**1**	3	0.35
	Coherence	**1**	2	3	0.049*
Fairy-tale	Resembles Genre	**1**	3	2	0.014*
	Coherence	**1**	3	2	0.013*

Table 8.4: Results of the full game evaluation participant study. *Indicates statistical significance ($p < 0.05$).

Genre	Questions	Random	Rules	Human
Mystery	Interesting	45	72*	69*
	Coherence	36*	45*	69*
	Resembles Genre	45	38*	75*
Fairy-tale	Interesting	42	37*	64*
	Coherence	25*	25*	45
	Resembles Genre	25*	37*	69*

8.2.2 Full Game Evaluation

This participant study was designed to test the overall game formulation process encompassing both phases described in Section 8.1. A single story from each genre was chosen by hand from the 10 stories used for the graph evaluation process. From the knowledge graphs for this story, I generate descriptions using the neural, rules, and random approaches described previously. Additionally, I introduce a human-authored game for each story here to provide an additional benchmark. This author selected was familiar with text-adventure games in general as well as the genres of detective mystery and fairy tale. To ensure a fair comparison, I ensure that the maximum number of locations and maximum number of characters/objects per location matched the other methods. After setting general format expectations, the author read the selected stories and constructed knowledge graphs in a corresponding three step process of: identifying the n most important entities in the story, mapping positional relationships between entities, and then synthesizing flavor text for the entities based off of said location, the overall story plot, and background topic knowledge.

Once the knowledge graph and associated descriptions are generated for a particular

story, they are then automatically turned into a fully playable text-game using the text game engine Evennia[2]. Evennia was chosen for its flexibility and customization, as well as a convenient Ib client for end user testing. The data structures were translated into builder commands within Evennia that constructed the various layouts, flavor text, and rules of the game world. Users were placed in one "room" out of the different world locations within the game they were playing, and asked to explore the game world that was available to them. Users achieved this by moving between rooms and investigating objects. Each time a new room was entered or object investigated, the player's total number of explored entities would be displayed as their score.

Each participant was was asked to play the neural game and then another one from one of the three additional models within a genre. The completion criteria for each game is collect half the total score possible in the game, i.e. explore half of all possible rooms and examine half of all possible entities. This provided the participant with multiple possible methods of finishing a particular game. On completion, the participant was asked to rank the two games according to overall perceived coherence, interestingness, and adherence to the genre. I additionally provided a required initial tutorial game which demonstrated all of these mechanics. The order in which participants played the games was also randomized as in the graph evaluation to remove potential correlations. I had 75 participants in total, 39 for mystery and 36 for fairy-tales. As each player played the neural model created game and one from each of the other approaches—this gave us 13 on average for the other approaches in the mystery genre and 12 for fairy-tales.

The summary of the results of the full game study is shown in Table 8.4. As the comparisons made in this study are all made pairwise between my neural model and one of the baselines—they are presented in terms of what percentage of participants prefer the baseline game over the neural game. Once again, as this is highly interrelated ranked data, I perform the Mann-Whitney U test between each of the pairs to ensure that the rankings

[2]http://www.evennia.com/

differed significantly. This is also indicated on the table.

In the mystery genre, the neural approach is generally preferred by a greater percentage of participants than the rules or random. The human-made game outperforms them all. A significant exception to is that participants thought that the rules-based game was more interesting than the neural game. The trends in the fairy-tale genre are in general similar with a few notable deviations. The first deviation is that the rules-based and random approaches perform significantly worse than neural in this genre. I see also that the neural game is as coherent as the human-made game.

As in the previous study, I hypothesize that this is likely due to the rules-based approach being more suited to the mystery genre, which is often more mundane and contains less fantastical elements. By extension, I can say that thematic commonsense in fairy-tales has less overlap with everyday commonsense than for mundane mysteries. This has a few implications, one of which is that this theme specific information is unlikely to have been seen by OpenIE5 before. This is indicated in the relatively improved performance of the rules-based model in this genre across in terms of both interestingness and coherence.The genre difference can also be observed in terms of the performance of the random model. This model also lacking when compared to my neural model across all the questions asked especially in the fairy-tale setting. This appears to imply that filling in gaps in the knowledge graph using thematically relevant information such as with AskBERT results in more interesting and coherent descriptions and games especially in settings where the thematic commonsense diverges from everyday commonsense.

Procedural world generation systems are required to be semantically consistent, comply with thematic and everyday commonsense understanding, and maintain overall interestingness. I describe an approach that transform a linear reading experience in the form of a story plot into a interactive narrative experience. My method, AskBERT, extracts and fills in a knowledge graph using thematic commonsense and then uses it as a skeleton to flesh out the rest of the world. A key insight from my human participant study reveals that the

ability to construct a thematically consistent knowledge graph is critical to overall perceptions of coherence and interestingness particularly when the theme diverges from everyday commonsense understanding.

CHAPTER 9
QUEST GENERATION

In this chapter, I consider the challenge of automatically generating narratives that have recognizable causal entailment between events, also known as quests. Specifically, I approach the problem of quest generation as story generation via *plot-infilling* (Ippolito *et al.* 2019; Donahue *et al.* 2020) where an outline of plot points is extracted from a source then elaborated upon. I introduce the concept of *soft causal relations*, where causal entailment between story events does not need to be strictly logically consistent, but draws upon people's everyday commonsense understanding of whether one event tends to be preceded or succeeded by another.

I demonstrate an approach to story generation using soft causal relations in the C2PO (Commonsense, Causal Plot Ordering) system, which generates narratives via plot infilling using soft causal relations. Inspired by work on plot graph learning (Li *et al.* 2012), C2PO attempts to create a branching space of possible story continuations that bridge between plot points that are automatically extracted from existing natural language plot summaries. To create this branching story space, I iteratively extract commonsense causal inferences from the COMET (Bosselut *et al.* 2019) model of commonsense reasoning. Finally, once the space—a plot graph—has been constructed, I search the space for complete sequences.

Using human participation studies, I evaluate C2PO against baseline text infilling systems with different uses of commonsense reasoning and inductive biases to determine the role of soft causal relations on perceptions of story quality. I choose two story corpora in different genres: real-world mystery stories such as Sherlock Holmes—known for generally being consistent with everyday commonsense norms, and children's fairy tales such as Hansel and Gretel—stories which usually shatter commonsense expectations. Through these studies I further explore the broader issue of how the change in commonsense norms

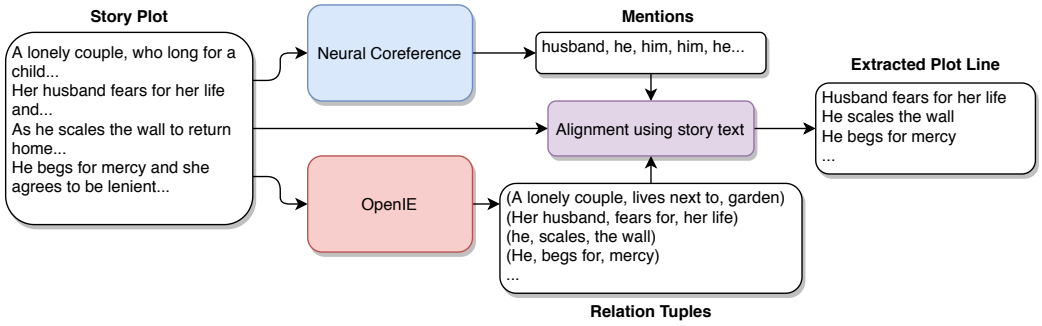


Figure 9.1: An illustration of high level plot point extraction.

across storytelling genres affects perceptions of story quality.

9.1 Soft Causal Relations

A *hard causal relation* implies that some world state transitions that are illegal—e.g., a character John cannot shoot Xavier if John is not in possession of a gun and the two characters are physically co-located. In contrast, a *soft causal relation* is mediated by the assumed reader's beliefs. Soft causality is therefore causality–normally a logical construct in narrative–mediated by the beliefs of the reader. It provides a causal ordering of events from the perspective of the reader instead of from the perspective of the author (whether human or agent). That is, a soft causal relation is a reasonable expectation of two non-mutually exclusive criteria: (a) certain activities are needed to achieve a character's goal, and (b) certain activities are in pursuit of future goals. The first clause draws on the psychological theory of the role of causality in story understanding by Trabasso and Broek (1985): readers attempt to understand "why" events occur by tracking causal relations as *enablement*—some event y cannot occur unless some preceding event x occurred. The second clause draws upon a theory of the role of character goal hierarchies in story understanding by Graesser *et al.* (1991): readers attempt to understand "why" things happen by tracking and predicting character goal hierarchies. In both cases, whether an inference is made by reader is strongly dependent on what the reader's beliefs about the world are. In short, the key difference between hard and soft causality is the idea of expectations of

causality via commonsense reasoning.

Commonsense knowledge is the set of commonly shared knowledge about how the world works. It enables us to form expectations about what will happen if I take certain courses of action and to infer things that likely happened in the past. Commonsense reasoning is the application of commonsense knowledge to specific contexts. Relevant to my work, commonsense reasoning might be applied to make inferences about what might have needed to have taken place for a character to arrive at a certain state—soft enablement—and what a reasonable next action would be based on what has happened so far—soft goal hierarchies.

Specifically for this paper, I use COMET (Bosselut *et al.* 2019) to model an assumed reader's commonsense knowledge. COMET is a transformer-based language model designed for commonsense inference and is trained on ATOMIC (Sap *et al.* 2019). ATOMIC is a dataset containing 877k instances of information relevant for everyday commonsense reasoning in the form of typed if-then relations with variables. ATOMIC is organized into different relation types such as "needs", "wants", "attributes", and "effects". I specifically use the relations for "wants" and "needs". An example of a cause using the *wants* relation is as follows, "if X tried to get away, then X *wants* to be free." Likewise, an example of an effect using the *needs* relation is, "if X scaled the wall, then X *needs* to know how to scale the wall."

The key difference between hard and soft causality is the idea of expectations of causality via commonsense reasoning and can be illustrated using the relations seen here. A hard causal relation requires verification and satisfaction of propositions, as in the example given in the paper - John cannot shoot Xavier if John is not in possession of a gun or they are not co-located. A soft causal relation here would be that the reader's belief that John dislikes Xavier and wants to fight him and thus as a result, he *wants* a weapon. Guns are weapons and thus there is a probability that John *needs* a weapon to fight Xavier.

In the next section I detail how I use the theory of soft causal relations, and COMET

commonsense inferences about needs and wants, to generate stories. In section 5, I present the results of a human participant study that uses an evaluation of several systems in two distinct genres to probe how soft causal relations affect participant perceptions of story quality and coherence.

9.2 C2PO

This section presents the overall layout of C2PO. C2PO works by first extracting a set of high level plot points from a given textual story plot S and then generating a branching set of events that go between each high level plot point. The final story is obtained by walking the overall plot graph generated by joining each generated sub-graph.

9.2.1 Plot Extraction

The overall plot extraction process is described in Figure 9.1. In order to facilitate plot extraction, I propose a method that uses coreference resolution and information extraction to identify a set of plot points following a single character. First, I extract all the coreference clusters using a pre-trained neural coreference resolution model (Clark and Manning 2016). There can be multiple such clusters, each of which contains all mentions in the story belonging to a single possible character. I pick one of these clusters at random. Let $M = \{m_1, m_2, ..., m_n\}$ denote this cluster. Simultaneously, I also extract a set $\mathcal{R}$ of $\langle subject, relation, object \rangle$ triples from the story text using OpenIE (Angeli *et al.* 2015).

Once I have both of the set of mentions for a character and the triples for the story, I align them, attempting to find the subset of triples $\mathcal{P} \subset \mathcal{R}$ that are relevant for a single character on the basis of their character-level positions within the original story text. Both the neural coreference model and OpenIE are information retrieval systems and so I can identify the character-level offset or position of the retrieved information in the original story text. Let $pos(\cdot)$ be a function that can do this. The set of plot points is $\mathcal{P} = \{\langle s, r, o \rangle : pos(m) = pos(s), \forall m \in M, \langle s, r, o \rangle \in G\}$. The result is a sequence of relational tuples in

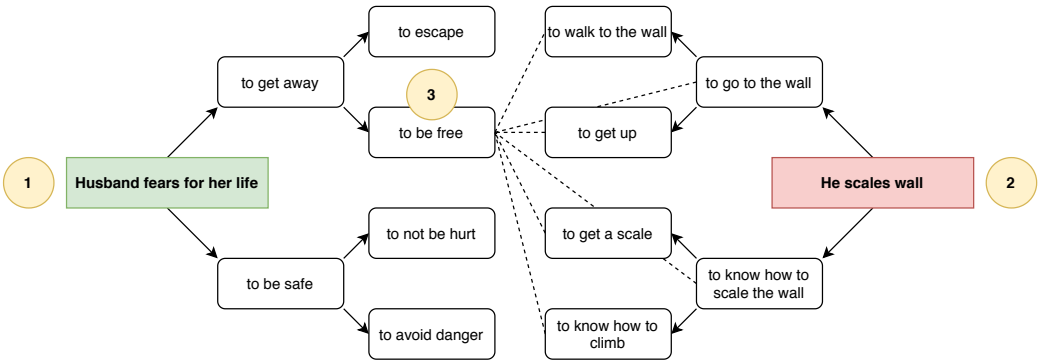


Figure 9.2: A demonstration of the plot graph generation process. 1 and 2 respectively indicate adjacent, extracted plot points. Dotted lines represent the process of finding the optimal link between the backward plot graph and node 3.

which the character is the primary subject of the triple, ordered by when they first appeared in the original story text. Joining each triple together yields a subject-relation-object phrase which I refer to as a *plot point*.

9.2.2 Plot Graph Generation

Once I have established a series of plot points $\mathcal{P} = \{p_1, p_2, ..., p_n\}$, I move on to plot graph generation as illustrated in Figure 9.2. A plot graph is generated for each pair of adjacent plot points $(p_i, p_{i+1}), i \in \{1, .., n-1\}$ and then linked together in the order the plot points first appear in P to form a plot graph for an entire story.

The process to generate a plot graph between adjacent plot points p_1, p_2 is as follows. Starting from p_1, I use COMET (Bosselut *et al.* 2019) to generate candidate next events in the story. The *wants* relation indicates a direct forward cause—a character has a want and therefore performs an action. I recursively query COMET to generate k event candidates n times going forward starting with p_1; let this be $\mathcal{G}^f$. The *needs* relation indicates backward enablement—a character needed something to be true to do an action. I recursively query COMET to generate k event candidate n times going backward from p_2; let this be $\mathcal{G}^b$. This gives us two directed acyclic graphs as seen in Figure 9.2. The relations in $\mathcal{G}^f$ and $\mathcal{G}^b$ are weighted proportional to the likelihood score produced by COMET for each inference.

Table 9.1: Dataset statistics.

	Mystery	**Fairy Tale**
No. Stories	569	695
Sentences per story	23.36	24.80
Vocabulary size	21,238	16,452

Table 9.2: Inductive biases of each system.

	Commonsense	**Storytelling**
C2PO	✓	✓
BERT+infill	✓	
Hier. Fusion		✓

The next step is to look for the optimal way to link $\mathcal{G}^f$ and $\mathcal{G}^b$ and computing the probability of reaching a node $u \in \mathcal{G}^f$ looking at all nodes $\forall v \in \mathcal{G}^b$. Let $Pr^{needs}(u|v)$ be the probability of generating event e_2 as determined by COMET under the $needs$ relation, conditioned on e_1, and $Pr^{wants}(v|u)$ be the same but under the $wants$ relation. I define this link's weight as:

$$w(u,v) = \frac{Pr^{wants}(u|v)}{\alpha_u^{wants}} + \frac{Pr^{needs}(v|u)}{\alpha_v^{needs}} \tag{9.1}$$

where α_u^{wants} and α_v^{needs} are normalization constants. Here I set them equal to the probability of generating the word "to", a word in ATOMIC common to both relation types. This process is repeated for all nodes until I have found a set of optimal links. [1]

Finally, I link together the plot graphs for the entire sequence of plot points: $\mathcal{G} = \bigcup_{p_1,p_2}(\mathcal{G}^f_{p_1} \cup \mathcal{G}^b_{p_2}), \forall p_1, p_2 \in \mathcal{P}$ where p_1, p_2 are adjacent in $\mathcal{P}$. A story can be generated via a random walk of the graph from the first plot point p_1 to the last p_n. All random walks are guaranteed to terminate in p_n because $\mathcal{G}^b_{p_n}$ is constructed by branching backward from p_n. Likewise, each intermediate plot point $p_2 ... p_{n-1}$ is a node in $\mathcal{G}$ that all walks must pass through.

Table 9.3: Examples of a story generated by each model in both genres given the same initial set of *bolded* high level plot points.

	C2PO	BERT+infill	Hierarchical Fusion
Mystery	**Holmes decides go.** Holmes wants to go. Holmes begins to see something. Holmes begins to look around. **Holmes notices a pair of trouser knees.** Holmes wants to clean up. Holmes begins take a shower. Holmes wants to get ready. Holmes wants to walk to the store. **Holmes taps in front of Wilson's shop.** Holmes tries say hello. Holmes wants start the car. Holmes tries to drive to the scene. **He calls Police Inspector Jones.**	**Holmes decides go.** Holmes new friend initially stays. Holmes new son accepts goes. Holmes mother also stays. **Holmes notices a pair of trouser knees.** Holmes himself still watches. Holmes again is house ghost watches. Holmes insists he took watch. **Holmes taps in front of Wilson's shop.** Holmes smiles and eventually leaves. Holmes red cap now appears. Holmes silhouette finally stands. **He calls Police Inspector Jones.**	**Holmes decides go.** The room was silent. The room was silent. The air was heavy , and the room was quiet. **Holmes notices a pair of trouser knees.** The young man wasn't going to be a father. His parents weren't supposed to be a father. They had the best kids in the entire world. **Holmes taps in front of Wilson's shop.** I'm not sure what's happening to me , but I'm not sure. What? You've been in a heel for a few years, and you've been in a heel for nearly a month. **He calls Police Inspector Jones.**
Fairy Tale	**Queen asks her mirror.** Queen wants to look better. Queen wants to try on clothes. Queen starts to be mad. **Queen is furious.** Queen tries to relax. Queen wants to take a nap. Queen starts to get up. Queen begins to approach someone. **She appears at a dwarfs'.** Queen starts to surprise everyone. Queen starts to have a party. queen wants to have money. Queen tries to buy poison comb. **She brushes with poisoned comb.** Queen tries to wash her hair. Queen starts dry it. Queen wants to be hungry. Queen wants to get the knife. **Queen cuts the apple in half.**	**Queen asks her mirror.** Queen is still half smiles. Queen who had had frowns. Queen has always asked. **Queen is furious.** Queen wife of mary then flees. Queen wife husband anna maria refuses. Queen mistress queen mistress wives demands. **She appears at a dwarfs'.** Queen queen rose meets princess. Queen sees fairies she crowns fairies. Queen rises with beauty. **She brushes with poisoned comb.** Queen was now also finally returns. Queen then had only disappears. Queen thought she vanished. **Queen cuts the apple in half.**	**Queen asks her mirror.** "What the ..." "You know I have no idea how I got here. You know I can't do anything about it." "I know I can't do anything about it." **Queen is furious.** A large, creature sits in the middle of a room with an odd looking cat on it. The creature is a strange looking cat, though it looks like the same cat is in its own room. Its fur is like a large, white slept. **She appears at a dwarfs'.** "So, you 're here to kill me," asked the man in the suit, with a slight hint of worry in his due. "Yes," replied the man in the suit. **She brushes with poisoned comb.** We hadn't met in a long time. We weren't supposed to be alone , and the rest of our group was just a group of people. **Queen cuts the apple in half.**

9.3 Experiments

I evaluate on a story dataset with two genres—mystery stories and fairy tales—first introduced in Chapter 8,[2] statistics for the dataset can be found in Table 9.1. The data is partitioned into train and test splits in a 8:2 ratio, and the train split used to train C2PO and two baseline models (described below). A random set of 10 stories is chosen from each genre in the test set and high level plot points are extracted as described in Section 9.2. For each model and for each set of high-level plot points and for each genre I generate three distinct stories for a total of $(3 \times 10 \times 2 \times 3 = 180$ stories. I generate three stories for each combination of model, plot point set, and genre to account for variance in stories that can be produced by the same high level plot due to the branching nature of C2PO as well as variance in the baselines' outputs. Standard automated language generation metrics such as perplexity and BLEU (Papineni *et al.* 2002) are known to be unreliable for creative generation tasks (Ammanabrolu *et al.* 2020c). The stories are thus evaluated using a human participant study, described below.

9.3.1 Baselines

I choose two baselines on the basis of the comparisons they afford (summarized in Table 9.2). Both are designed to perform text infilling tasks but differ based in their inductive biases. "Inductive biases" here specifically refer to a system's ability to model commonsense knowledge and if they were originally designed for storytelling or not.

BERT+infill. The first baseline is a BERT (Devlin *et al.* 2019) based model that has not strictly been designed for storytelling (though BERT is trained on a corpus that includes story texts) and then adapted to perform text infilling. Although large-scale pre-trained language models are known not to be great storytellers, mostly due to them being unable

[1]COMET and ATOMIC can be replaced by any model designed for automated knowledge base completion and corresponding commonsense reasoning knowledge base by selecting the appropriate relations in the replacements.

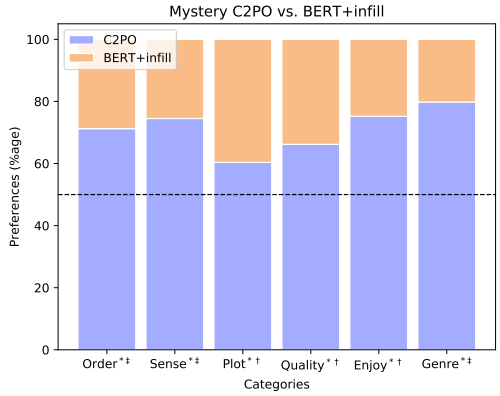
[2]https://github.com/rajammanabrolu/WorldGeneration

Table 9.4: Participant count statistics.

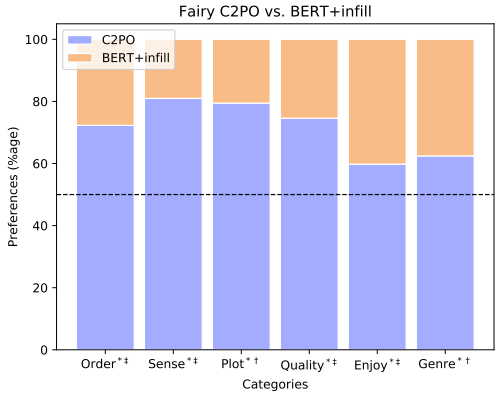
	C2PO vs. BERT+	C2PO vs. Hier.	Tot.
Mystery	82	89	171
Fairy Tale	90	90	180
Total	172	179	351

to stay on track for any extended period of time (See *et al.* 2019), they have demonstrated knowledge of factual commonsense information by virtue of the amount of data they have been trained on (Petroni *et al.* 2019). Our problem setting requires us to generate a section of text between two consecutive high level plot points at a time, reminiscent of approaches taken by Ippolito *et al.* (2019) and Donahue *et al.* (2020) that condition a language model on left and right contexts to fill in blanks in a story. I follow a similar setup for this baseline, using BERT (Devlin *et al.* 2019) conditioned to attend to both previous tokens—the preceding plot point—and future tokens—the following plot point—to generate sequences (Lawrence *et al.* 2019). BERT+infill is fine-tuned using this methodology on the high-level plot points extracted from my training data. Despite being similar to these prior methods, I note that BERT+infill utilizes no storytelling domain knowledge in its architecture and boils down to simple masked language modeling with multiple mask tokens.

Hierarchical Fusion. Fan *et al.* (2018) train their system—consisting of a convolutional sequence-to-sequence network with self-attention (Ott *et al.* 2019)—on the Reddit Writing Prompt corpus, where human-contributed prompts are paired with human-contributed stories. The system learns to first generate a prompt and then transform it into a story. This model's architecture is explicitly designed to tell stories and is suited for a type of storytelling wherein a prompt for a story is generated into a passage This type of training is particularly well suited to my setup of generating a story piece-by-piece using extracted high level plot points. I train the model from my training set using high level plots extracted from the stories as described in Section 9.2 as the prompts and sections in between each of these extracted plot points as the story.



(a) C2PO vs BERT+infill in the mystery genre.



(b) C2PO vs BERT+infill in the fairy genre.

Figure 9.3: Human evaluation results comparing C2PO vs BERT+infill. * indicates $p < 0.05$, ‡ indicates $\kappa > 0.4$ or moderate agreement, † indicates $\kappa > 0.2$ or fair agreement

9.3.2 Human Evaluation Setup

I have 10 sets of high level plots per genre and three generated stories per each plot for each of the models. I recruited 351 human participants via Mechanical Turk. Criteria for enrollment included: (a) fluency in English, and (b) demonstrating an understanding of commonsense based causality in stories. To screen participants for the latter I asked them to predict potential next events that could reasonably occur given a story scenario.

Human participants are given one story generated by C2PO and another evenly randomly picked from those generated by either BERT+infill or Hierarchical Fusion for the same plot. The order that these stories are presented in is randomized to account for bias induced due to the ordering effect (Olson and Kellogg 2014). Each story pairing is seen by at least three participants. Participant count statistics are given in Table 9.4.

Participants are then asked a series of questions, each measuring a particular aspect of perceived story quality, comparing the C2PO generated model to one of the baselines. For each question they are asked to note down which story they preferred. The questions I use are adapted from Purdy *et al.* (2018) and have been used in multiple storytelling works as an indication of story quality (Ammanabrolu *et al.* 2020c; Tambwekar *et al.* 2019). Specifically, I ask:

Table 9.5: Statistics for generated stories. Unique n-grams are measured with respect to those found in the test set of the initial story data.

	C2PO		**BERT+infill**		**Hierarchical**	
	Myst,	Fairy	Myst,	Fairy	Myst,	Fairy
Avg. Sent/Story	29.23	30.2	25.4	26.0	31.3	41.0
Avg. Words/Sent	4.94	5.04	4.62	4.79	7.21	5.75
Unique Bigrams	312	317	356	357	380	402
Unique Trigrams	1245	1353	1856	1870	2187	2190

1. Which story's events occur in a more PLAUSIBLE ORDER?: as a proxy to indicate perceptions of overall causality within the story.

2. Which story's sentences MAKE MORE SENSE given sentences before and after them?: to examine perceptions of local causality and commonsense reasoning in the story.

3. Which story better follows a SINGLE PLOT?: for insight into perceptions of global coherence for the entire story.

4. Which story is of HIGHER QUALITY?: as a measure of overall perceived story quality.

5. Which story is more ENJOYABLE?: indicates story value.

6. Which story better FITS A GENRE?: as a measure of how well the story matches commonsense knowledge specific to a genre, capturing the differences between the two genres.

For each of these questions, within a pairwise comparison, I perform a paired Mann-Whitney U test to assess statistical significance and additionally calculate Fleiss' κ (Kappa) value to measure inter-rater reliability.

9.4 Results and Analysis

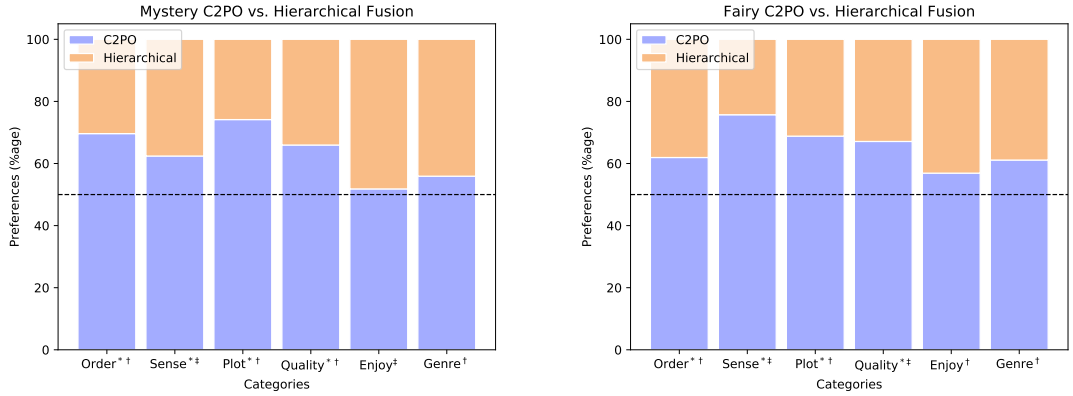
There are a few dimensions along which I will attempt to analyze these results: (1) the inherent inductive biases of each model as seen in Table 9.2, (2) the two genres, and (3) the

Table 9.6: Randomly selected examples of stories generated by the fairy models. Bolded sentences are the original extracted plot points.

C2PO	BERT+infill	Hierarchical Fusion
They live out at time seven years. Bearskin village is just came. Bearskin and family wants country. Bearskin only takes river gives valley. Bearskin gave purse of gold. Bearskin brothers also agreed. Bearskin with senior chief heads agreed. Bearskin daughter among elder sisters agrees. Bearskin promised return in three years. Bearskin they had had promised. Bearskin daughter was sisters agreed. Bearskin daughter sister married my family. **Her sisters ridiculed her.** Bearskin also always once reappeared. Bearskin who always is later left. Bearskin also becomes a. **Bearskin found devil again At end of seven years.** Bearskin says and so says. Bearskin he can sing now sings. Bearskin it has told him cries. **He fulfill his promise.** Bearskin polish grease nail nails. Bearskin polish cut boot. Bearskin helps clean burn wood cuts. **Bearskin clip his nails.** Bearskin boots and mr. Bearskin boots nails and boot. Bearskin leather leather toe boot. **He is good.** Bearskin brother had still also stood. Bearskin looked and then said. Bearskin claimed it did. **Bearskin dropped his half of ring.**	**They live out at time seven years.** Bearskin village is just came. Bearskin and family wants country. Bearskin only takes river gives valley. Bearskin gave purse of gold. Bearskin brothers also agreed. Bearskin with senior chief heads agreed. Bearskin daughter among elder sisters agrees. Bearskin promised return in three years. Bearskin they had had promised. Bearskin daughter was sisters agreed. Bearskin daughter sister married my family. **Her sisters ridiculed her.** Bearskin also always once reappeared. Bearskin who always is later left. Bearskin also becomes a. **Bearskin found devil again At end of seven years.** Bearskin says and so says. Bearskin he can sing now sings. Bearskin it has told him cries. **He fulfill his promise.** Bearskin polish grease nail nails. Bearskin polish cut boot. Bearskin helps clean burn wood cuts. **Bearskin clip his nails.** Bearskin boots and mr. Bearskin boots nails and boot. Bearskin leather leather toe boot. **He is good.** Bearskin brother had still also stood. Bearskin looked and then said. Bearskin claimed it did. **Bearskin dropped his half of ring.**	**They live out at time seven years.** He was standing here in a large room. The door was wide open , and he looked down. He was n't sure what he was seeing , but he was n't sure what to think.Bearskin gave purse of gold. " Hey , buddy , " the man said , " You 've got to be kidding me. " The man laughed , " Yeah , man. " " I 've been watching this guy for a long time , buddy.Bearskin promised return in three years. " What do you mean I 'm here for you. " " Well , " said the young boy , " I 'll have you know this is n't a bad idea. " " But , " said the young girl , " I was n't in any trouble.**Her sisters ridiculed her.** " What the hell. " He was a man in a suit of black suit with a black tie. His face was red with sweat and a scar that was almost like lightning.**Bearskin found devil again At end of seven years.** " Hey , man , what the Dom are you doing. " I said as I looked around. " I 'm not sure , man , " he said with an expression of ease , " you 're not really my man.**He fulfill his promise.** The man was n't exactly the most spinning man in the world. He was n't the most spinning man in the world. He was n't the most spinning man in the world , but he was n't the most famous man in the world.**Bearskin clip his nails.** I was n't the best on this wonders. I did n't need any. There is n't much time.**He is good.** " So , what are you doing here. ""‘ Oh , it 's been a while since you 've been in this business. ""‘ I do n't really know much about this.**Bearskin dropped his half of ring.**
Prince calls at time night. Prince and tries again calls. Prince that still wakes. Prince then knocks and asleep. **He finds To his horror.** Prince robert francis charles george leaps. Prince charles henry louis rupert. Prince joseph john frederick maurice victor. **He leaps from tower.** Prince anthony or saint john leaves. Prince anthony nicholas edward lawrence. Prince edward nicholas james peter george. **He wanders For years.** Prince james edward thinks. Prince james john james. Prince alexander rupert james augustus george. **He hears again her voice.**	**Prince calls at time night.** Prince and tries again calls. Prince that still wakes. Prince then knocks and asleep. **He finds To his horror.** Prince robert francis charles george leaps. Prince charles henry louis rupert. Prince joseph john frederick maurice victor. **He leaps from tower.** Prince anthony or saint john leaves. Prince anthony nicholas edward lawrence. Prince edward nicholas james peter george. **He wanders For years.** Prince james edward thinks. Prince james john james. Prince alexander rupert james augustus george. **He hears again her voice.**	**Prince calls at time night.** He was sitting alone in the dark , surrounded by the many stars. He was alone , and this was it. He had been waiting for this moment , a long time ago.**He finds To his horror.** " I 'm not really sure where I got that from , but I just want to know where I got it " " Oh , well it 's not a problem , you have to take it , just like I did " " But no it 's not , it 's not like you could just give it to me. I 'm not going to take it , but I do n't want to be like it. It 's not like you have any idea why " " You 're right , it 's not like you have the time.**He leaps from tower.** " So ... " " What. " " I 'm not going to ask you any more questions about this. " " You 're going to be fine.**He wanders For years.** The sound of his phone being hung off the ground made him look up and see the phone. He had n't seen her since she was eight. She was the most beautiful thing he 'd ever seen.**He hears again her voice.**
Girl has clothes. Girl lives in dress. Girl know she can dress. Girl know still know she sees. **Her wear rags.** Girl not smile and talk. Girl never gets bad hair. Girl and most rarely smiles. **She do kinds of hard work.**	**Girl has clothes.** Girl lives in dress. Girl know she can dress. Girl know still know she sees. **Her wear rags.** Girl not smile and talk. Girl never gets bad hair. Girl and most rarely smiles. **She do kinds of hard work.**	**Girl has clothes.** The first few days were strike as hell. The first few days I woke up on a cold , damp night. My eyes were Small and my face looked as if I had been taller , bleeding , bleeding , I was just a little bit sprinting and confused by the pain.**Her wear rags.** The room was dark. Not a single light was on. All the lights in the room were on.**She do kinds of hard work.**

Table 9.7: Randomly selected examples of stories generated by the mystery models. Bolded sentences are the original extracted plot points.

C2PO	BERT+infill	Hierarchical Fusion
Holmes decides go. Holmes wants to go. Holmes begins to see something. Holmes begins to look around. **Holmes notices has trouser knees.** Holmes wants to clean up. Holmes begins take a shower. Holmes wants to get ready. Holmes wants to walk to the store. **Holmes taps in front of Wilson 's shop.** Holmes tries to say hello. Holmes tries say hello. Holmes wants start the car. Holmes tries to drive to the scene. **He calls Police Inspector Jones.**	**Holmes decides go.** Holmes new friend initially stays. Holmes new son accepts goes. Holmes mother also stays. **Holmes notices has trouser knees.** Holmes himself still watches. Holmes again is house ghost watches. Holmes insists he took watch. **Holmes taps in front of Wilson 's shop.** Holmes smiles and eventually leaves. Holmes red cap now appears. Holmes silhouette finally stands. **He calls Police Inspector Jones.**	**Holmes decides go.** The room was silent. The room was silent. The air was heavy , and the room was quiet.**Holmes notices has trouser knees.** The young man was n't going to be a father. His parents were n't supposed to be a father. They had the best kids in the entire world.**Holmes taps in front of Wilson 's shop.** " I 'm not sure what 's happening to me , but I 'm not sure. " " What. " " You 've been in a heel for a few years , and you 've been in a heel for nearly a month.**He calls Police Inspector Jones.**
Alec has room. Alec tries to sleep. Alec wants take a shower. Alec begins to get dressed. Alec begins go to room. **Alec to room.** Alec tries to eat. Alec starts drink water. Alec wants to have money. Alec wants to have a car. **Alec throttle Holmes.** Alec starts to be successful. Alec tries to buy a car. Alec tries to go to the car. Alec wants to be in a car. **His father apparently twisting Holmes 's wrist.** Alec begins to hurt someone. Alec tries to do something bad. **Alec out of hand.**	**Alec has room.** Alec back to says back says. Alec thinks back to goes. Alec well that sure did too. **Alec to room.** Alec sees and also sees baldwin. Alec himself sees baldwin waits. Alec herself eventually enters. **Alec throttle Holmes.** Alec fletcher holmes thomas john thomas. Alec watson james smith. Alec james stewart john hacking. **His father apparently twisting Holmes 's wrist.** Alec getting out suddenly went outside. Alec said i always hesitated. Alec really only half even laughed. **Alec out of hand.**	**Alec has room.** I had been sitting in this room for a long time. I had never met a man before , but I had n't been there when I was in here. I was not sure why I was in here.**Alec to room.** " What do you mean , it 's not real. ! " " Oh no. No.**Alec throttle Holmes.** This is not my first time writing. I 'm in a bit late for this so it 's not the first time I 've written anything but I 'm not going to start it. I hope I did n't mess up this.**His father apparently twisting Holmes 's wrist.** " Hey you , " said refuge. " What. " " What 's this thing.**Alec out of hand.**
Wilder hired Hayes. Wilder begins to give orders. Wilder wants to follow up. Wilder begins to hear news. **Wilder heard news.** Wilder tries to learn more. Wilder starts to do well. Wilder wants to work hard. He confessed all. Wilder begins to go home. Wilder begins to sleep. Wilder wants to get ready. Wilder begins to go to the restaurant. **He let his younger son stay at inn.** Wilder starts to go to bed. Wilder tries wake up. Wilder wants to work. Wilder begins to have money. **James Wilder seek his fortune.**	**Wilder hired Hayes.** Wilder s brothers family initially agreed. Wilder s had resigned. Wilder sr announced d v. **Wilder heard news.** Wilder actually really cried. Wilder so alone has really wept. Wilder himself who only found sobs. He confessed all. Wilder and he refused. Wilder again is threatened. Wilder i again himself insisted. **He let his younger son stay at inn.** Wilder story was b. Wilder horror story by w. Wilder werewolf tale mr. **James Wilder seek his fortune.**	**Wilder hired Hayes.** " What is this. " he asked , as he walked to the door. A door with a large metal door that was like an egg.**Wilder heard news.** The tree was still , the tree 's spirits was a tree 's tree. The tree was still , the tree , its tree and the tree were still. The tree was still , a tree , its tree and its tree and its tree.He confessed all. He walked into the bar and took a seat. He took a long , long drag of the cigarette. " What have I done , " he asked , " You have to stop me , " and he leaned forward to take another puff.**He let his younger son stay at inn.** The man looked at me and smiled. The man looked at me and said , " You 're my only child , " he said , " I 'm sure your father is n't a bad man , " he said. " I do n't think I have the right to be like you , " I said.**James Wilder seek his fortune.**
Colonel has behaviour. Colonel begins to get better. Colonel wants to get up. Colonel starts to go to the door. Colonel wants to walk to the lock. **He would lock himself.** Colonel begins to get in the car. Colonel starts to drive. Colonel begins to drink. Colonel wants get drunk. **He shouting in drunken with pistol.** Colonel tries to sleep. Colonel begins to get up. Colonel wants to go outside. Colonel wants to go to garden. **He was found dead in garden pool.**	**Colonel has behaviour.** Colonel was a must saw. Colonel is has did. Colonel not that was thought. **He would lock himself.** Colonel charles brown was. Colonel thomas and james a. Colonel thomas edward l. **He shouting in drunken with pistol.** Colonel general henry william miller killed. Colonel william andrew wilson acting. Colonel james edward richard stirling died. **He was found dead in garden pool.**	**Colonel has behaviour.** " You 're kidding me. " I shouted. " You 're joking about that.**He would lock himself.** " I 'm sorry sir , but we do n't have the time. " " You did n't do this. " " We 're not here for that.**He shouting in drunken with pistol.** I 've been on this planet for three years. It 's been a few weeks , and it 's been quite some time since I 've been here. I 'm here.**He was found dead in garden pool.**



(a) C2PO vs. Hierarchical Fusion in the mystery genre.

(b) C2PO vs. Hierarchical Fusion in the fairy genre.

Figure 9.4: Human evaluation results comparing C2PO vs, Hierarchical Fusion. * indicates $p < 0.05$, ‡ indicates $\kappa > 0.4$ or moderate agreement, † indicates $\kappa > 0.2$ or fair agreement

questions asked of the participants. The analysis will be performed hierarchically in the order just presented. Table 9.5 provides statistics on generated stories and Table 9.3 displays select examples of generated stories for each of the models in both genres. Tables 9.6, 9.7 provide qualitative examples of stories by randomly selected plots from first the fairy tale, then the mystery genre.

9.4.1 C2PO vs BERT+infill

Figures 9.3a and 9.3b show the percentages that participants preferred C2PO versus the BERT+infill system for each dimension and for each story genre. C2PO is preferred over BERT+infill in both genres and in all dimensions. All of these results are statistically significant ($p < 0.05$) with fair-to-moderate inter-rater reliabilities.

For the mystery genre the greatest differences in preferences are observed with respect to enjoyability and genre resemblance. The systems were most similar with regard to their ability to maintain a single plot. For the fairy tale genre the greatest differences are seen in terms of the story events' plausible ordering, making sense causally, and the ability to maintain a single plot. The models were most similar with regard to their genre resemblance and enjoyability.

The questions that C2PO does particularly well on compared to BERT+infill are complementary across the genres. Enjoyability and genre resemblance are rated higher for C2PO in the mystery genre as opposed to fairy tales. I additionally observe that these two factors are highly, positively correlated using Spearman's Rank Order Correlation ($r_s = 0.56, p < 0.01$). Similarly, C2PO performed comparatively better in terms of plausible ordering, making sense causally, and the ability to maintain a single plot for fairy tales than for mysteries. These three factors are also highly, positively correlated with each other and in terms of overall perceived story quality ($0.6 > r_s > 0.55$, $p < 0.01$ for all pairwise comparisons).

This provides evidence that the brand of commonsense reasoning-based causality brought to bear by C2PO—needs and wants—works well in the mystery genre. The mystery genre follows everyday commonsense norms whereas the fairy tale genre is more likely to stray from commonsense norms. It can thus be inferred that genre-specific or thematic commonsense knowledge is required to improve perceptions of genre resemblance and enjoyability but does little in terms of metrics assessing local and global coherence in terms of causality.

9.4.2 C2PO vs Hierarchical Fusion

Figures 9.4a and 9.4b show the percentages of participants that preferred C2PO to Hierarchical Fusion. For the mystery genre, C2PO was preferred for the dimensions of plausible ordering, making causal sense, maintaining a single plot, and overall story quality. These dimensions were significantly different ($p < 0.05$). The dimensions of enjoyment and genre resemblance were not significantly different, meaning no system did better than the other.

I see a similar pattern for fairy tale stories: C2PO is preferred to hierarchical fusion for the same dimensions as the mystery genre and are not significantly different for enjoyment and genre resemblance.

Across genres, there is a positive correlation between metrics relating to coherence and

overall perceived story quality ($0.6 > r_s > 0.5$, $p < 0.05$ for each pairwise comparison using Spearman's Rank Order Correlation). Also recall that the Hierarchical Fusion model contains an inductive bias for storytelling but does not model commonsense reasoning. This appears to indicate that genre resemblance and enjoyability are not dependant on causal, commonsense reasoning but rather on the how much the generated text "sounds like a story" but story quality still depends on overall coherence.

9.4.3 Broader Trends

There are two main trends that one can see across the models depending on their inductive biases (extent to which the models are trained for commonsense reasoning or storytelling). I observe these trends on the basis of the analysis presented so far as well as the examples of output stories found in Table 9.3. (1) Having commonsense reasoning abilities generally improves perceptions of local and global coherence in terms of causality with a caveat that what is perceived as commonsense can change across genres. When genre or domain specific commonsense knowledge matches "everyday" commonsense, it makes for an automated storyteller that is significantly more causal in nature. (2) Just commonsense reasoning without any sort of storytelling inductive bias incorporated—such as with pre-trained and finetuned language models which themselves have no real penchant for storytelling—into a model's design doesn't help, however, in terms of enjoyability and genre resemblance. The performance of Hierarchical Fusion in terms of enjoyability and genre resemblance—and the examples seen in Table 9.3—appear to indicate that models designed for storytelling do a better job of maintaining the writing style of a story but struggle with causality.

9.5 Conclusions

I intend for the findings of this work to be utilized by researchers studying quest generation as automated storytelling, a standing AI grandchallenge requiring creative, long-form

language generation. Most prior works in the area either incorporate inductive biases in the form of either commonsense reasoning abilities or storytelling domain knowledge and do not measure the impact of these biases across a wide set of human perceived metrics relating to story quality. I explore the effects of *soft causal relations*—reasonable expectations by a reader regarding a story's progression—on human-based perceptions of overall story quality. I introduce C2PO as a way to use *soft causal relations* via transformer-based models trained for commonsense inference in storytelling.

A key insight from a human participant study, measuring a wide set of human perceived metrics, shows that the sum of the parts is indeed greater than the whole. Automated storytellers require both domain specific commonsense reasoning abilities as well as a storytelling inductive bias incorporated into the design of the system to perform well in terms of: local and global coherence on the basis of causality, enjoyability, genre resemblance, and overall story quality. Further, perceptions of causal, commonsense conforming coherence are highly correlated with overall story quality. I encourage authors of future work to build on these findings and more closely explore lines of research that use thematically relevant *soft causal relations* to improve automated quest and story generators.

CHAPTER 10
PUTTING IT ALL TOGETHER

In this chapter I teach goal-driven agents to interactively act and speak in situated environments by training on generated curricula—specifically focusing on bringing together the two lines of my research to create agents that generalize more effectively to unseen environments and scenarios. My agents operate in LIGHT (Urbanek *et al.* 2019), building on much of the work seen in Chapter 7. Goals in this environment take the form of character-based quests, consisting of personas and motivations. I augment LIGHT by learning to procedurally generate additional novel textual worlds and quests to create a curriculum of steadily increasing difficulty for training agents to achieve such goals. In particular, I measure curriculum difficulty in terms of the rarity of the quest in the original training distribution—an easier environment is one that is more likely to have been found in the unaugmented dataset. An ablation study shows that this method of learning from the tail of a distribution results in significantly higher generalization abilities as measured by zero-shot performance on never-before-seen quest data.

A core machine learning problem standing in the way of robust generalization of situated agents is the ability for such agents to adapt to environments that are novel with respect to their training environments. The distribution of all possible situations that can be encountered by an agent has a long tail, with many scenarios being encountered very rarely. In sequential decision making problems in particular, this generalization gap is the result of an agent simply memorizing trajectories, e.g. the sequence of actions and dialogues required to finish a game, and thus being unable to react in novel scenarios. One way of decreasing this generalization gap is by training agents on procedurally generated environments—wherein the agent learns a family of parametrized tasks with a significantly larger state-action spaces than singular environments, thus effectively making the memo-

rization of trajectories impossible (Justesen *et al.* 2018; Cobbe *et al.* 2020).

Drawing inspiration from all of these ideas, I create a method that creates and augments textual environments by creating a curriculum of increasingly more difficult procedurally generated novel environments, and corresponding goals, for agents to learn from in a manner that removes the long tail from the distribution. I measure curriculum difficulty in terms of the rarity of finding a goal-environment pair in the original training distribution—an easier environment is one that is more likely to have been found in the unaugmented dataset.

As seen in Figure 10.3, I use LIGHT (Urbanek *et al.* 2019), a large-scale crowdsourced fantasy text-adventure game, consisting of a set of locations, characters, and objects possesses rich textual worlds, but without any notion of goals to train goal-driven agents. This is further extended by Chapter 7 who introduce dataset of quests for LIGHT and demonstrations of humans playing these quests, providing natural language descriptions in varying levels of abstraction of motivations for a given character in a particular setting. To complete these quests, an agent must: (1) maintain character via its persona; and (2) reason in a *partially observable* world about potential actions and utterances based on incomplete descriptions of the locations, objects, and other characters. This requires several human like competencies such as commonsense reasoning, dynamic natural language understanding, and operating in combinatorially sized language-based state-action spaces.

Our contributions are threefold: (1) I present a method of parametrizing and jointly generating a curriculum of goal-environment pairs in LIGHT; (2) I show how to effectively train reinforcement learning agents on this curriculum; and (3) Provide an experimental study showing that my method enables significantly better generalization than those training on singular environments.

Setting	You are in the Dangerous Precipice. The dangerous precipice overlooks the valley below. The ground slopes down to the edge here. Dirt crumbles down to the edge of the cliff. There's a dragon crescent, a knight's armor, a golden dragon egg, and a knight's fighting gear here. A knight is here.You are carrying nothing.
Partner:	Knight.
Persona	I am a knight. I come from a lower-ranking noble family. I serve under the king, as my father did before me. In times of war, I fight on horseback.
Carrying	knight's armor, golden dragon egg, knight's fighting gear
Self:	A dragon.
Persona	I am a dragon living in the mountains. I enjoy hoarding treasure. I terrorize the local populace for fun.
Carrying	Nothing.

Figure 10.1: Setting and character information for both self and partner characters as taken from LIGHT.

Motivations:		**Timeline:**	
Short	I need to recover the dragon egg that was stolen and punish the knight.	-4 hours	go to dangerous precipice
		-15 min	get knights armor from knight
		-10 min	get golden dragon egg
Mid	I need to return the golden dragon egg to my treasure hoard.	Now	hit knight
		+5 min	put dragon egg on back
		+15 min	eat the knight
Long	I need to build the largest hoard ever attained by any one dragon.	+2 hours	go to the mountains

Figure 10.2: Motivations with different levels of abstractions and corresponding sequence of timeline actions in chronological order for the self character in LIGHT-Quests. There are 7486 quests in total.

Insssssolent pessst! I should immolate you for this tressssspasss.

And why is that, dragon?

Ssstealing my precciousss golden egg! I'll tell you what, I'll give you 10 sssseconds to amussse me with your sssstory and THEN I'll burn you alive!

You said you wanted to attack me, dragon, did you not?

Go ahead, I'm lisssssstening.
get golden dragon egg

Now now! I would have given you that had you asked!

Asssssssk for my own property back? What a riduculousss notion

Look here, I told you to watch your mouth and you didn't, so leave or I'll make you leave.

And now threatsss! Thisss is proving to be a mossst engaging conversssation.
hit knight
Give my regardsss to the valley floor below!

Figure 10.3: Example of a demonstration of a human (blue shaded) completing the above quest while role-playing as the self character with a partner agent (grey shaded). There are 2111 such human demonstrations of average sequence length 12.92, consisting of 22672 dialogues in total.

10.1 LIGHT-Quests Recap

Recall that LIGHT game environment (Urbanek *et al.* 2019) is a multi-user fantasy text-adventure game consisting of a rich, diverse set of 1775 characters, 663 locations, and 3462 objects. Characters are able to perform templated actions to interact with both objects and characters, and can speak to other characters through free form text dialogues. Actions

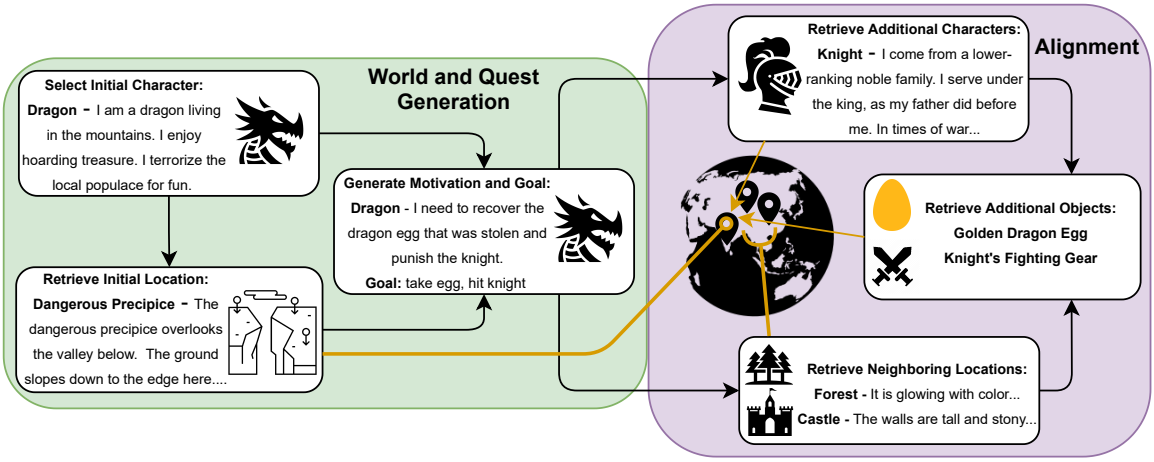


Figure 10.4: Procedural environment generation pipeline. Black lines indicate conditioning on all prior components. Gold lines indicate (adjacent) location placement.

in text games generally consist of verb phrases (VP) followed optionally by prepositional phrases (VP PP). For example, *get OBJ, put OBJ, give OBJ to CHAR*, etc.. These actions change the state of the world which is expressed to the player in the form of text descriptions.

Quests in LIGHT (Chapter 7) take the form of motivations in three levels of abstraction—short, mid, and long—corresponding to differing amounts of the timeline. The short motivation is always guaranteed to correspond most closely the sequence of actions that reach the world state required to finish the game. For example, if the short motivation is to acquire a sword, then the corresponding goal state would be for the character to have a sword in their inventory. There are 5982 training, 756 validation, and 748 test quests. This environment also contains a set of human expert demonstration of people speaking and acting in character while playing one of the quests mentioned above—giving us sequences of actions and dialogues needed to finish the game. The average sequence of a human demonstration is 12.92, with an average action sequence length of 2.18 and dialogue of 10.74. There are 1800 training, 100 validation, and 211 test human expert demonstrations after the data was filtered.

10.2 Procedural Environment Generation

This section describes my procedural generation pipeline as seen in Figure 10.4, starting with world and quest generation, followed by aligning both of them. There are two main kinds of models that I use for the different modules in this pipeline: retrieval and generative.

Retrieval models are trained to return the most highly correlated output for a given input in the dataset. For example, a retrieval model can be asked to return the most likely character that can be found at a particular location. These models compare a human annotated gold standard label with negative candidates drawn from the dataset. The negative candidates provide noise that the model must filter out in order to learn representations that let it best predict the gold label. These models are generally trained via a ranking loss that maximizes the scores of the gold label while simultaneously minimizing negative candidate score. At test time, the highest ranked candidate based on the score is selected as the model prediction.

The generation-based models used in this pipeline are trained to return the most likely output sequence given an input sequence. Given a target sequence $Y = \{y_1, ..., y_M\}$ and some input context via the encoders X. These models use autoregressive decoding techniques that factor the distribution over the target sequence into a chain of conditional probabilities with a causal left to right structure as $P(Y|X;\theta) = \prod_{i=1}^{M+1} p(y_i|y_{0:i-1}, X;\theta)$ where θ represents the current network parameters. At test time, a special start-of-sequence token is provided to the model which then proceeds to decode the rest of the output sequence using beam search.

Additional training details for all models mentioned in this section are found in Appendix B.4.7.

10.2.1 World and Quest Generation

The first step of the pipeline involves choosing an initial character who will perform the quest. For this, I uniformly randomly sample from the set of characters found in the LIGHT-Quest training set. The corresponding character information includes a name and a textual description of the character's persona.

Given this character information, I further retrieve the location that that character is most likely to be found in. I use a retrieval-based ranker model that checks for similarity of StarSpace (Wu *et al.* 2018) embeddings. Our choice of model is influenced by Fan *et al.* (2019) who report state-of-the-art retrieval performance for locations in LIGHT using this model. The overall ranker model first trains a randomly initialized StarSpace embedding model that is designed to correlate characters with the locations they are found in. It learns a single bag-of-words embedding that takes into account all the individual words contained within the input—encoding character and location information as well as the previously mentioned negative retrieval candidates. The rest of the training is similar to other retrieval models described earlier. The retrieved location information consists of a location name as well as a textual description of the location.

The quest is now generated using the existing character and location information. I train two BART (Lewis *et al.* 2020) models that encodes input information via a bidirectional transformer encoder and decodes autoregressively: the first takes as input character and location information and produces a short motivation (Section 10.1); the second takes as input character, location information, short motivation and produces the sequence of LIGHT game engine executable actions needed to achieve the motivation. This sequence of actions is provided by the human expert demonstrations as mentioned in Section 10.1.

10.2.2 Aligning Worlds and Quests

At this stage, the environment contains a motivated main character to perform a quest and a location for them to start in. I now focus on aligning the world with the quest to ensure that

the quest is playable and achievable. Intuitively, to ensure that a quest is achievable, the world needs to contain all of the entities—locations, characters, and objects—mentioned within the quest.

To this end, the alignment process involves training three BERT-based (Devlin *et al.* 2019) biencoder retrieval models to retrieve the most likely characters, locations, and objects required flesh the environment out and make the quest achievable. I use the same biencoder architecture proposed by Urbanek *et al.* (2019) which encodes context using one transformer and candidates with another—scoring candidates via inner product between the two encoded vectors. The character retrieval model is conditioned on the initial character, quest, and location—producing additional characters required to complete the world. For LIGHT RL, I follow the setup in Chapter 7 and restrict worlds to only contains 2 characters at maximum but note that this method is extendable to greater numbers of characters. Similarly, the location retrieval model is also conditioned on the same things—producing, in this case, 4 neighbors to the initial location (resulting in worlds that are 5 locations large). These locations are connected to the initial location and a character can move between them by using commands such as *go west*, *go up* etc.. Once these characters and locations are added to the world, the object retrieval model predicts the set of objects that are required to be distributed for each location given all the character information present in it. The final game environment instance is complete once this object set has been added to the game.

10.3 Curriculum Learning

This section first describes the base reinforcement learning setup for training motivation-driven agents in LIGHT and then explores the topic of curriculum generation.

10.3.1 Reinforcement Learning

Following the definition of text game POMDPs (Chapter 2), we see that the LIGHT environment further allows us to factorize the overall action space $\mathcal{A}$ into A as the set of possible

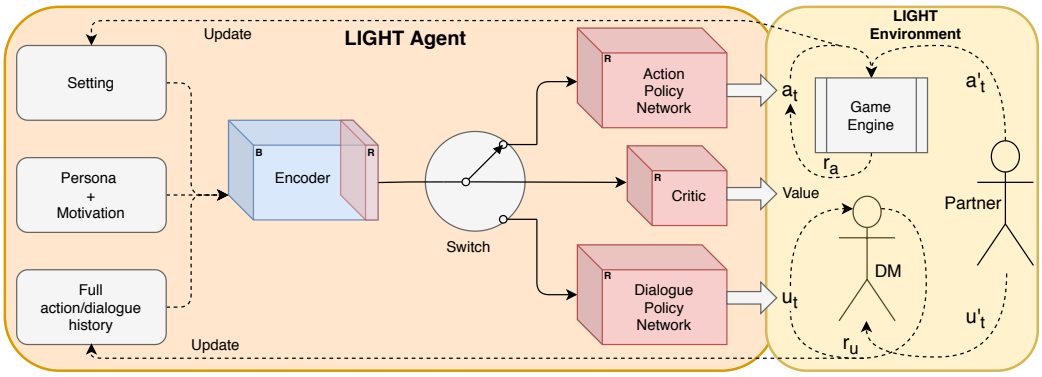


Figure 10.5: Overall architecture and training pipeline for the LIGHT RL Agent.

textual actions or commands (e.g. *get sword, steal coins from merchant*), and U as the set of possible dialogues that can be uttered by an agent, thus making it a factored POMDP (Degris and Sigaud 2013). This in turn means that, for a given quest q, each expert human demonstration $\mathcal{D}(q) = \alpha_0^*, \alpha_1^* ... \alpha_n^*$ can be factorized into two sub-sequences of expert demonstrations of actions and dialogue $\mathcal{D}_A(q) = a_0^*, a_1^*, ... a_n^*$ and $\mathcal{D}_U(q) = u_0^*, u_1^*, ... u_m^*$ respectively. The factorized action spaces A and U are constructed by enumerating all possible actions/dialogue utterances in the all human demonstrations in LIGHT-quests—$A = \bigcup_{q \in Q} \mathcal{D}_A(q); U = \bigcup_{q \in Q} \mathcal{D}_U(q)$ with $|A| = 4710$ and $|U| = 22672$.

10.3.2 Generating Curricula

I generate curricula by building off of my procedural LIGHT game instance generation pipeline. I make the observation that the original quests in LIGHT are heavily skewed towards certain quest types—with the majority involving goals and short motivations that contain objectives related to getting and object, and hitting or hugging another character (Figure 10.6). I further note that the first verb in the short motivation forms the basis of the quest for that particular agent.

Actions in LIGHT, and more generally in text games, are executed in the game engines on the basis of verbs—engine subroutines are linked to verbs with nouns forming arguments—and as such are primarily responsible for changing the state of the world. For

example, *get sword* invokes the *get* subroutine that places an object, in this case a sword, in the character's surrounding into their inventory. As the quest is generated early in the pipeline, with the world and the rest of the components being conditioned on it, I can say that the first verb in the short motivation is an important dimension along which I can assess the distribution of individual LIGHT game instances. Thus, concretely, the verb counts from the short motivation aggregated over a set of quests represents the primary dimension along which I measure the distribution of quests.

Parametrizing Curriculum Difficulty As LIGHT-Quests is crowdsourced, it exhibits an expected but significantly visible imbalance—as seen in Figure 10.6—in the quest types that are available for the agent to train on. Given this relative imbalance of this multinomial distribution, I hypothesize that a LIGHT agent only learns to do well on certain types of objectives and not others—memorizing trajectories for less seen quest types, i.e. those found in the tail of the distribution. Preliminary evidence for this hypothesis is also seen in Prabhumoye *et al.* (2020), where they show a positive correlation between the number of instances of a particular type of quest during training and the final test goal-achievement performance. Based on these observations and my initial hypothesis, I use this particular dimension to *parametrize curriculum difficulty* for training LIGHT agents—quest types that are rarer in the initial training data will be harder for the agent to generalize to in a zero-shot setting.

Intuitively, I seek to create curricula that contain a diverse set of game instances with quest types that are not often found in the initial training data. Our earlier observations let us hypothesize that this will enable the LIGHT agent to more effectively learn from rare instances of quests as opposed to memorizing the corresponding trajectories. To this end, the generated curricula each consist of a pool of quests with steadily decreasing quest type imbalance. In my case, this imply that the flatness of the multinomial distribution increases until it tends towards being uniform with respect to the categorical quest type variable.

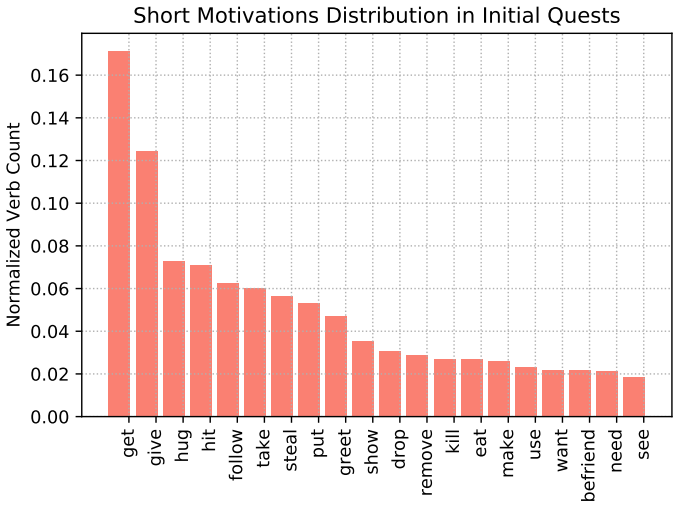


Figure 10.6: Normalized top-20 verb count distribution of short motivations of the original LIGHT-Quests dataset.

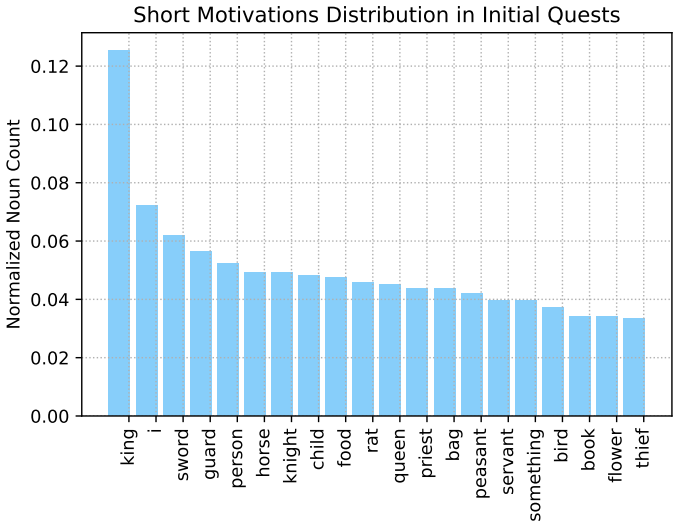


Figure 10.7: Normalized top-20 noun count distribution of short motivations of the original LIGHT-Quests dataset.

This is done by running the procedural generation pipeline iteratively until the number of instances for the highest count quest type is within n of the lowest count quest type. The total number of additional generated instances is held fixed across curriculum, only the distribution of quest types within each curriculum changes.

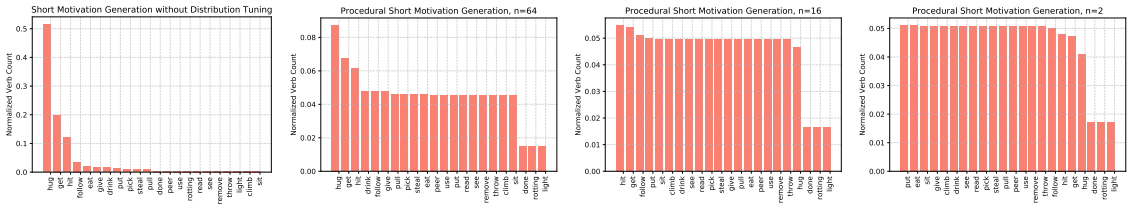


Figure 10.8: Top-20 distribution of verbs in the short motivation of the curriculum of quests starting from the original generated curriculum on the left to the flattened, **generated** curriculum on the right as a function of n (Section 10.3.2). The y-axis of the different verbs reflect their normalized overall count in the pool of quests.

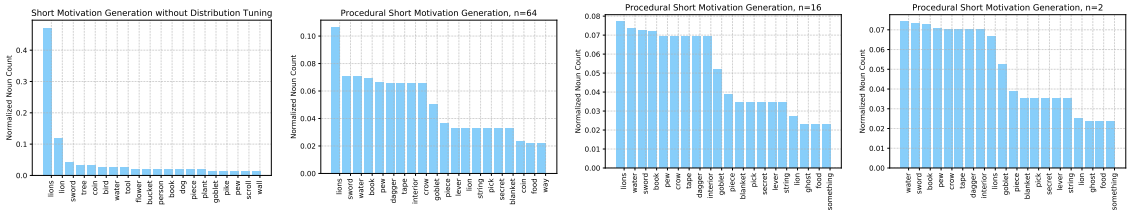


Figure 10.9: Top-20 distribution of nouns in the short motivation of the curriculum of quests starting from the original generated curriculum on the left to the flattened, **generated** curriculum on the right as a function of n (Section 10.3.2).

Figure 10.8 shows that decreasing n has the intended effect of decreasing imbalance with respect to verb types. Generating using this pipeline has the added effect of increasing diversity within the pool of each available quest type. One measure of diversity within the pool of a single quest type is the types of nouns contained within the short motivations—these generally correspond to the characters, locations, and objects mentioned. Figure 10.9 shows that decreasing imbalance in the verb types for a short motivation also results in decreasing imbalance in noun types, once again corresponding to decreasing n. Short motivation generation is one of the first steps in the pipeline, i.e. the rest of the pipeline is conditioned on it, and as such increasing the flatness of the distribution there has the effects of increasing distribution for downstream components.

10.3.3 Training

Figure 10.5 shows the overall architecture and training pipeline—our reinforcement learning pipeline is unchanged from that shown in Chapter 7 with the exception of the curriculum of quests performed by the agent and the way the speech rewards are designed. An encoder first takes in information about setting, persona, motivation for a single character then passes it onto a switch module. This switch module is a meta policy that decides if an agent should act or talk and is trained to mimic how often human experts act or talk while performing quests via demonstrations. Two separate policy networks make a decision on which action to perform or dialogue to say given the current context and a single shared critic attempts to measure the value of taking an action in a particular state.

Once an agent acts or talks, the partner agent—in this case also a polyencoder (Humeau *et al.* 2020) trained to react to agents with motivations—also acts or talks and this information is processed by the environment. As recommended by Ammanabrolu *et al.* (2021) and Prabhumoye *et al.* (2020), I keep the partner model fixed during the episodes where the LIGHT agent trains to ensure that it retains natural English semantics—avoiding the problem of language drift by learning an emergent language with that must agree with the partner's usage (Lee *et al.* 2019).

Rewards. All actions, either those of the agent-in-training or the partner agent, are processed by the engine, checking for goal state completion—hence known as *act goals*. For example, if the LIGHT agent had the motivation to acquire a sword, the goal could be completed via a: *self act completion*: where the agent acquires a sword itself by picking it up, stealing it, convincing the partner to drop theirs so you can pick it up, etc. *partner act completion*: where the agent uses speech to convince their partner to achieve the goal for them (e.g., by persuading the partner to give them the sword).

Following Chapter 7, I use a learned model–the Dungeon Master (DM)—to score the agent's ability to speak. The DM used here is a poly-encoder model trained on collected human quest demonstrations as well as the original conversations in LIGHT. It is conditioned

on quests and motivations and thus able to provide a (noisy) indication of how natural the agent's dialogue utterances are given its immediate context, similarly to the function of the DM during the data collection process.

Given the dialogue portion of a human quest demonstration $\mathcal{D}_U(q) = u_0^*, u_1^*, ...u_n^*$, of length n, the DM returns a reward r_u of $\frac{1}{2n}$ if an utterance was in the demonstration $u \in \mathcal{D}_U(q)$ (for a maximum of one time per episode for each utterance from the demonstration). A further $\frac{1}{2n}$ is given each time the utterance is scored as being within the top-k most likely utterances by the DM. The original quests all have human demonstrations but the procedurally generated ones do not. During training, in cases where a particular LIGHT game instance does not have corresponding human demonstration, only the latter reward resulting from an utterance being within the top-k most likely utterances by the DM is used. This naturalness objective will be hence referred to as a *speech goal*. These rewards thus also denser than *act goals*, helping the agent learn overall. Further, similarly to the game engine, the DM also provides a set of M valid utterances which are the M most likely dialogue candidates from the candidate set for the current context.

A2C Curriculum Training. Overall training is done via A2C (Mnih *et al.* 2016) a policy gradient algorithm that maximizes long-term expected reward by comparing the advantage $A(s_t, a_t^*)$ of taking an action in a state to the average value of taking a valid action as predicted by the critic $V(s_t)$. Each parallel A2C agent samples from the the current pool of available quests—i.e. the curriculum—for a fixed number of steps k before switching to the quest pool corresponding to the next higher level difficulty curriculum. The initial pool of quests is the training set of LIGHT-Quests as seen in Chapter 7 and all pools after that correspond to decreasing values of n used when generating the curricula (as seen in Figure 10.8). Further training details are identical to Chapter 7.

Table 10.1: Procedural generation evaluation showing metrics for each individual model in the pipeline.

Pipeline Step	Model	Hits@10	F1	Ppl
World Generation				
Location	Biencoder	0.543	0.153	-
Object	Biencoder	0.563	0.154	-
Character	Starspace	0.653	0.289	-
Quest Generation				
Short Motivation	BART	-	0.488	7.55
Goal Action	BART	-	0.763	3.75

10.4 Evaluation

I conduct two separate evaluations: the first measures the effectiveness of the various models in the procedural environment generation pipeline as well as the effectiveness of the pipeline as a whole; the second provides zero-shot ablations of the LIGHT RL agents trained on the resulting curricula.

10.4.1 Procedural Generation Evaluation

All of the models in the pipeline described in Section 10.2 are trained using only the training set of the original LIGHT and LIGHT-Quests data. LIGHT-Quests inherits characters, locations, and objects from the original LIGHT dataset and adds on motivations and goals in the form of quests. Thus, the character, location, and object retrieval models are evaluated on the LIGHT unseen test set and the motivation and goal generation models are evaluated on the LIGHT-Quests test set. I report the standard array of metrics: hits@10 and F1 ranking prediction score for retrieval models; and F1 (as a harmonic average of BLEU-1 (Papineni *et al.* 2002) and ROUGE-1 (Lin 2004)) and perplexity for generative models. Hyperparameters for all models are found in Appendix B.4.7.

Analysis. Table 10.1 presents the results of this evaluation. There are two primary trends to note: (1) character retrieval is easier than retrieving location and objects—likely due to the ; and (2) goal action generation is easier than motivation generation. I hypothesize

that the first trend is a direct consequence of the fact that generated motivations and goals regularly contain the names of the characters involved but mostly leave implicit information such as the objects required—e.g. the action *hit dragon* as a knight would require a weapon such as a sword to be equipped first. The second trend stems from the fact that goal actions can often be thought of as condensed version of the short motivation—number of tokens required to generate goal actions is far less than short motivations. This implies that the goal action model is akin to a summarization model as opposed to the short motivation model which has the more difficult task of generating the motivation with only initial character persona and location information.

10.4.2 Curriculum Learning Evaluation

This evaluation tests the LIGHT RL agent's ability to zero-shot generalize to unseen environments. Agents were each zero-shot evaluated on 211 human demonstrations from the LIGHT-Quests test set for a single episode per quest across three independent runs. The study ablates across how the curricula are generated, for the best two model types found in Chapter 7. The two model types are:

Scratch. No pre-training is done, the encoder is a 3-layer randomly initialized transformer and trained along with the policy networks.

Adaptive. Pre-training is done on the tasks introduced in Chapter 7 by training a 12 layer transformer with 256 million parameters using a cross-entropy loss as seen in Humeau *et al.* (2020). These weights are then transferred to the **B**lue shaded portion of the encoder as seen in Figure 10.5 and frozen. A further three randomly initialized-layers are appended on to the end, indicated by the **R**ed portions, into which gradients flow.

A brief description of the tasks follows: Encoders are first trained on both pushshift.io Reddit and the commonsense dataset ATOMIC-LIGHT, giving the agent general priors on how to act and speak. The parameters are then tuned via multi-task trained using: (1) the same tasks as before to provide additional regularization; and (2) all tasks in LIGHT-

original and LIGHT-Quests, giving the agent priors on how to act and speak with motivations in the LIGHT fantasy domain.

Sampled Curricula. Inspired by Graves *et al.* (2017) and Chawla *et al.* (2002), I explore an alternate method of creating curricula by simply oversampling the same rare quests found in the tails of the distributions. This method does not generate new environments via the pipeline, instead choosing to sample rarer instances of quests with a higher weight when initializing each parallel A2C actor. This means that, effectively, the distribution of verbs looks similar to what it is in Figure 10.8 but the quests within a pool are repeated multiple times and are less diverse. The sampling process is weighted such that the distribution is determined by parameter n. This parameter means the same as it does in the procedurally generated environments seen in Section 10.3.2, i.e. the range in counts of various quest types based on the primary verb within.

For each model type and curriculum generation method, I present results on an agent's ability to act and speak when: (1) trained without a curriculum or any form of distribution tuning; (2) trained with a pool of quests that have been distribution tuned as seen in Figure 10.8 as a function of n; and (3) trained with a curriculum on steadily increasing, distribution tuned difficulty quest pools. For the first two methods, an agent received 10^7 total environment interactions per parallel A2C agent in a batch of 16. For the curriculum learning method, the agent received 2.5×10^6 interactions per pool of quests starting with the initial pool of untuned quests and then sequentially with $n = 64, 16, 2$ resulting in a total of 10^7 total environment interactions per parallel A2C agent in a batch of 16.

Analysis. Table 10.2 presents the results of this evaluation. I first report that the overall proportion of a pool of generated environments that contain achievable quests or goals for a single curriculum is 0.89. This metric provides a proxy for measuring the accuracy of the alignment process and the overall error rate of the pipeline. The high achievability rate means that only a small proportion of LIGHT RL A2C agents will waste environment in-

Table 10.2: Curriculum learning evaluation. All experiments were averaged over 3 random seeds. Standard deviations across any individual result do not exceed 0.02. The "All Goals" column refers to quests where the agent has simultaneously achieved both types of goals within the allotted one episode.

Expt.	Act Goals	Speech Goals	All Goals
Scratch Encoder			
No Curr.	0.418	0.118	0.103
Sampled			
only n=64	0.392	0.113	0.097
only n=16	0.431	0.116	0.099
only n=2	0.435	0.124	0.111
curriculum	0.460	0.145	0.138
Generated			
only n=64	0.426	0.121	0.107
only n=16	0.433	0.129	0.112
only n=2	0.432	0.130	0.112
curriculum	**0.477**	**0.163**	**0.155**
Adaptive Encoder			
No Curr.	0.420	0.330	0.303
Sampled			
only n=64	0.431	0.336	0.312
only n=16	0.45	0.340	0.317
only n=2	0.456	0.339	0.321
curriculum	0.473	0.358	0.344
Generated			
only n=64	0.445	0.341	0.330
only n=16	0.469	0.367	0.359
only n=2	0.471	0.366	0.357
curriculum	**0.506**	**0.382**	**0.373**

teractions learning from quests that cannot be completed—increasing this rate even further would likely also improve the RL agents' sample efficiency.

The significantly increased performance of the generated curricula over the sampled curricula also indicates the importance of diversity within a single quest type. The sampled quests contain multiple instances of the same quest type but the generated ones have higher variability—leading to an increased observation space, ensuring that the agent cannot simply memorize trajectories.

Further, I see that just the distribution tuning by itself shows no significant gains in performance over the baselines trained on the original data and in fact loses performance in certain cases. In contrast, learning from the individually tuned quest pools in a sequential curriculum increases performance significantly. This appears to indicate that LIGHT

RL agents need to be trained with quests pools of steadily increasing difficulty—starting immediately on a set of quests with a high proportion of rare, generated quests has the potential to degrade performance.

We'd finally to note that the adaptive pre-trained model takes advantage of the generated curricula and distribution tuning more than the non-pre-trained scratch encoder, showing consistenly higher performance across the board. I hypothesize that this is likely a consequence of the adaptive model having greater model capacity—the pre-training enabling it to learn generalizable representations of the generated environments.

10.5 Conclusions

I focused on the problem of improving zero-shot generalization abilities of goal-driven RL agents to act and speak via natural language. An (obviously) key component of achieving this is to train the RL agents on a balanced training dataset that matches the distribution of the test data. As this is an unlikely scenario in most real-world applications, I make the observation that we can artifically augment a pool of training environments by generating curricula to mimic this—drawing on many of the conclusions made in prior chapters on both game play and game generation. This involves making an assumption regarding what dimension(s) you wish to measure the distribution on and further regarding what kind of distribution you expect the test data to be. In this domain, with goal-driven situated natural language agents, I hypothesize—and gather supporting evidence suggesting—that an effective way to parametrize such distributions is by looking at the primary verbs within an agent's motivation and bringing the distribution of verb types as close to uniform as possible. Despite minor errors in intermediate steps during procedural environment generation, training LIGHT RL agents via generated and distribution tuned curricula significantly improves their ability to generalize to unseen scenarios and environments.

10.6 Limitations and Future Work

This section will primarily discuss the limitations of knowledge graphs as applied to interactive narrative environments. Some of these limitations include questions pertaining to applicability outside this domain and modality—leading to a discussion of a couple of contemporary applications for the core technologies developed in this area.

10.6.1 Limitations of Knowledge Graph-based State Representations

Knowledge graphs in the RDF triple form, in return for the increased interpretability of a human readable state representation, are unable to represent probabilities and other continuous domains well without some additional work. Probabilities arise generally due to uncertainty or stochasticity in the environment—e.g. at any given step there is a 20% probability that a character X will be in a location Y and that they will move the rest of the time. When this is represented in knowledge graph form—i.e. as the agent explores the world—the agent represents this as a certainty depending on the observation. This means that if the agent sees character X at location Y, it now believes with 100% certainty that this is where the character is located even after the agent has moved away from the location. It will maintain this belief until it receives an observation contradicting this belief, at which point it will update the knowledge graph. Similar cases arise when using knowledge graphs for generating text games as well.

This is mitigated via datasets like JerichoWorld which contain multiple transitions showing character X at various locations—training over all of these locations can give models like the Worldformer an idea that there's a chance that this character can appear in one of n locations but still does not attempt to explicitly model this in the decoded knowledge graphs. Similarly, applying graphs to continuous domains requires the domains to be discretized—e.g. real-world robotics environments into demarcated locations. A key problem going forward lies in handling and modeling such uncertainty explicitly in the

symbolic graph form or perhaps through dynamic pre-training on large scale knowledge graphs as seen in the Light-RL encoders.

Further, overall utility and value of knowledge graph-based representations is made clear in tasks requiring long term coherence. Two examples of this, as seen in this work, are sequential decision making—operating in interactive environments—and structured language generation such as game generation and storytelling. The utility of the knowledge graph decreases rapidly for tasks requiring only limited horizon of context, or in which all the required context is provided such as some static unstructured question-answering datasets like SQuAD (Rajpurkar *et al.* 2016). In such cases, it may be counter-productive to have an intermediate persistent memory.

10.6.2 Transfer Across Domains and Modalities

Many of the core challenges presented by text games manifest themselves across domains with different modalities and it may be possible to transfer progress between the domains.

Interactive narratives provide tractable, situated environments in which to explore highly complex interactive grounded language learning without the complications that arise when modeling physical motor control and vision—situations that voice assistants such as Siri or Alexa might find themselves in when improvising responses. The abilities of agents that I've developed here, especially the LIGHT-RL agent, would directly transfer to creating policies for Alexa style chatbots to better engage with a user by generating language that is more contextually relevant and better dependent on long term memory of the conversation.

Let's now consider situations where we move away from text only situations. Take the example of a slice-of-life walking simulator text game where the main quest is to complete a recipe as given before. What happens when we encounter a similar situation with the added modality of vision? Can we take the knowledge we've gained from learning a text-based policy by completing the recipe in the original text game and use that to learn how to do something similar with a visually embodied agent? ALFWorld (Shridhar *et al.* 2021) tests

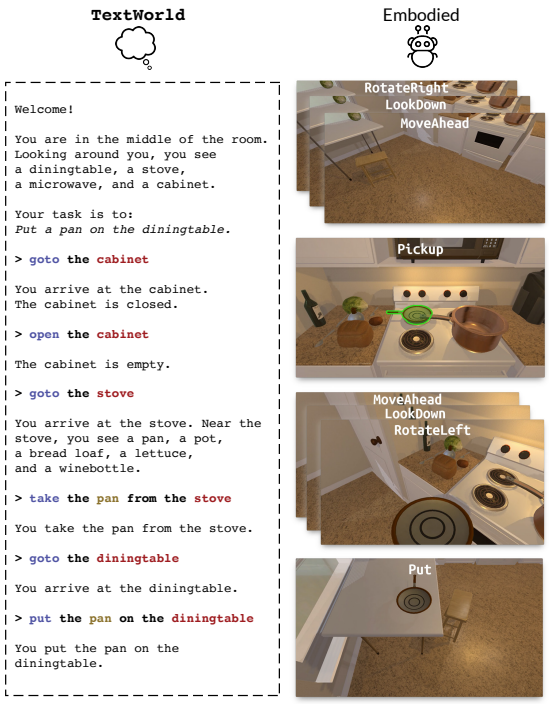


Figure 10.10: An aligned text game and visual environment from *ALFWorld* (Shridhar *et al.* 2021).

this idea, it is a simulator that lets you first learn text-based policies in the "home" text-game TextWorld (Côté *et al.* 2018), and then execute them in similarly themed scenarios from the visual environment ALFRED (Shridhar *et al.* 2020). They find that commonsense priors—regarding things like common object locations, affordances, and causality—learned while playing text-games can be adapted to help create agents that generalize better in visually grounded environments. This indicates that text games are suitable environments to train agents to reason abstractly through text which can then be refined and adapted to specific instances in an embodied setting.

Another such cross-domain transfer experiment was tested in the X-WLP dataset Tamari *et al.* (2021), where they collected and buil a corpus of complex wet lab biochemistry protocols that are framed as a quest and could thus be executed via a text-game engine. The annotations themselves are collected using a text-game-like interface, reducing overall data collection cost. Prior works (Tamari *et al.* 2019) discuss automatically extracting these protocols from raw lab texts and also training deep reinforcement learning agents on the

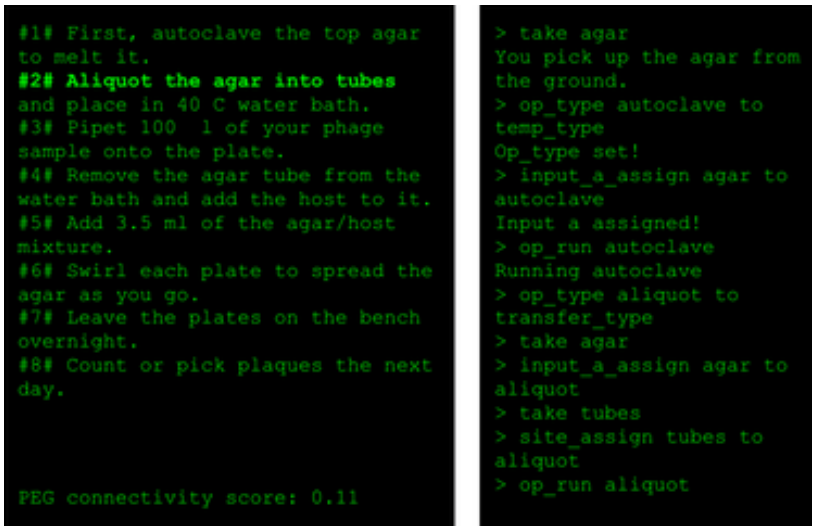


Figure 10.11: A wet lab protocol as a text game from the X-WLP dataset (Tamari *et al.* 2021).

resulting text-game quest. The ability to automatically frame wet lab experiments in the form of text game quests and leverage the latest text-game agent advances to interactively train agents to perform them has implications for significantly improving procedural text understanding (Levy *et al.* 2017) and in the reproducibility of scientific experiments (Mehr *et al.* 2020).

Appendices

APPENDIX A
WORLD MODELING EXPERIMENTS

This Appendix first provides additional samples for the dataset for qualitative purposes and then provides training details for the baseline models.

A.1 JerichoWorld Dataset

The games used in the Jericho suite and here are all open sourced freeware. The walkthroughs required to create the oracle agents for the collection of data for the games were drawn from various sources on the internet and errors were corrected manually. I provide our data at https://github.com/JerichoWorld/JerichoWorld under an MIT license. I provide 3 samples drawn from different games in the full dataset to help the readers better understand the diversity of text there.

```
Game: 905
Location:
   Bedroom (in bed)
   This bedroom is extremely spare, with dirty laundry scattered haphazardly all over the
       floor. Cleaner clothing can be found in the dresser. A bathroom lies to the south,
       while a door to the east leads to the living room.

   On the end table are a telephone, a wallet and some keys.

   The phone rings.
Observation: You take off the gold watch. The phone rings.
Inventory:
   You are carrying:
     some soiled clothing (being worn)
     a gold watch
Prev Act: take off watch
Inventory Objects:
   gold watch: Apparently it's 9:07. The phone rings.
```

soiled clothing: These clothes are a lost cause, sad to say no amount of laundering is going to get these stains out.

Inventory Attributes:

watch: animate, equip

clothing: animate, equip

Surrounding Objects:

phone: An ordinary telephone, notable chiefly for being fifteen or twenty years old.

keys: House keys, car keys, they're all on the same chain.

end table: A small end table, oak veneer over plywood.

living room: The living room lies to the east.

dirty dresser: Just a simple dresser.

laundry: Shirts, pants, the usual.

floor, east, south: You see nothing unexpected in that direction. The phone rings.

wallet: It's a brown leather wallet.

door: Just a regular door.

Surrounding Attributes:

keys: animate, equip

wallet: animate, equip

Graph: [you, have, gold watch], [you, in, bed], [you, have, soiled clothing]

Valid Actions: take phone, get up, take off clothing, take off watch, take keys, close door, take wallet, close door, put clothing down, put watch down, put clothing on table, open wallet, put watch down, put clothing on phone, put watch on table, put gold on phone, look under bed

Game: deephome

Location:

Secret Entrance

This is a rather dark and small room, having only two exits, back north the way you came, from the ancestral homes of Tana, or through the heavily barred wooden door before you that leads southwest and inward to the abandoned Deephome, abode of the Dwarves in Telleen. It has been three hundred years since your people lived here.

The heavy door stands open, admitting you into Deephome.

Observation: As you touch the finely etched symbol, you hear a click and a whir. Then the door swings open before you, opening into the abandoned city of Deephome. Your score has just gone up by five points.

Inventory:

You are carrying:

King's Order

a lantern (providing light)

Prev Act: push mountain

Inventory Objects:

lantern: This is an old and trusty (not rusty) lantern that has been in your family for centuries. It has yet to shut off at an inopportune moment. However, there is a saying in your family..."That lantern is bound to go off at an inopportune time sometime!"

order: The note reads: "Reclaimer: You have the esteemed duty to return to our Mountain Kingdom of Deephome and prepare it for our return. There are several things a Reclaimer must do: 1. Restore Power to the City 2. Restore Water to the city. 3. Visit each location and make sure it is safe, a quick appraisal should be sufficient. 4. Open the City Gates once more. 5. MOST IMPORTANT: Make sure the city is SAFE to return to. May the Peace of Kraxis go with you King Derash of the Mountain Tana, the year 782 SK."

Inventory Attributes:

lantern: equip

Surrounding Objects:

southwest: You see nothing special about the southwest wall.

house: It is the typical human house, maybe two stories. It is etched into the wood.

wooden door: This door is made of thick and sturdy wood. It has three symbols on it, a tree, a house, and a mountain.

symbols: On the door there are pictures of a mountain, a tree, and a house.

tree: The tree symbol looks as if it were etched into the wood.

mountain: The mountain looks mighty, a high peak among the clouds. It is etched into the wood.

Surrounding Attributes:

door: unlockable

symbols: unlock

Graph: [symbols, in, Secret Entrance], [wooden door, in, Secret Entrance], [ground, in, Secret Entrance], [you, in, Secret Entrance], [house, in, Secret Entrance], [Kraxis, in, Secret Entrance], [you, have, lantern], [mountain, in, Secret Entrance], [you, have, "Kings Order"], [tree, in, Secret Entrance]

Valid Actions: say manaz, push mountain, close wooden, get in southwest, put light down, put order down

Game: reverb

Location:

Behind the Counter

You are behind the counter at "Mr. Tasty’s Pizza Parlor". To the southwest is the rest of the restaurant.

On the counter is a large pizza box (which is closed).

You can see a handwritten note here.

Observation: You put the large pizza box on the counter.

```
Inventory: You are carrying nothing.
Prev Act: push large to counter
Inventory Objects:
Inventory Attributes:
Surrounding Objects:
   southwest: You see nothing special about the southwest wall.
   handwritten note: The note reads: "Stanley, Don't forget to make your delivery to Mr.
       Calzone, located at the San Doppleton Courthouse. You're already on thin ice, kid.
       One more screwup and you can expect to be looking for a new job." The note is
       signed with the initials "RT". The paper is official "Mr. Tasty's" stationery with
       the name Bob "Tasty" Tasker and lots of balloons and smiley faces all over the
       border. Isn't that cute?
   large pizza box: It's a large, flat, greasy cardboard box. Hastily scrawled on the
       outside is the word "Calzone". Which is weird, because it's clearly a pizza.
   counter: It's a majorly boring counter which you're unfortunately very familiar with.
Surrounding Attributes:
   handwritten note: indoor, readable
   large pizza box: indoor
   counter: indoor
Graph: [metal file, in, large pizza], [you, in, Behind the Counter], [handwritten note, in
    , Behind the Counter], [large pizza, in, large pizza box], [counter, in, Behind the
    Counter], [large pizza box, in, counter]
Valid Actions: get up, take note, take large, examine note, undo large, push note to
    southwest, push large to southwest, push note to counter, push large to counter
```

A.2 Baselines

The Rules and QA systems, are trained using hyperparameters and methodologies described in their respective chapters.

Rules

The exact details regarding knowledge graph updates are found as follows. At every step, given the current state and possible attributes as context. The rest of the triples are extracted using OpenIE (Angeli *et al.* 2015).

- Linking the current room type (e.g. "Kitchen", "Cellar") to the items found in the

room with the relation "has", e.g. $\langle kitchen, has, lamp \rangle$

- All attribute information for each object is linked to the object with the relation "is". e.g. $\langle egg, is, treasure \rangle$
- Linking all inventory objects with relation "have" to the "you" node, e.g. $\langle you, have, sword \rangle$
- Linking rooms with directions based on the action taken to move between the rooms, e.g. $\langle Behind\ House, east\ of, Forest \rangle$ after the action "go east" is taken to go from behind the house to the forest

Question-Answering

The QA models are trained on the SQuAD 2.0 (Rajpurkar *et al.* 2018), the Jericho-QA text game question answering dataset on the same set of training games as found in Worldformer, and then on Worldformer itself by formatting our dataset in the style of questions and answers when possible. Our dataset is formatted in the style of Jericho-QA by templating questions that ask about location, objects (including characters), and attributes. An example of a Worldformer dataset example converted to Jericho-QA format is seen below—though I would like to note that this removes much of the information present naturally within our dataset. All other model architecture and hyperparameter details are as seen in KG-A2C.

```
Game: reverb
Location:
   Behind the Counter
   You are behind the counter at "Mr. Tasty's Pizza Parlor". To the southwest is the rest
       of the restaurant.

   On the counter is a large pizza box (which is closed).

   You can see a handwritten note here.
Observation: You put the large pizza box on the counter.
Inventory: You are carrying nothing.
Question: Where am I located? Answer: Behind the Counter
```

```
Question: What is here? Answer: large pizza box, handwritten note, southwest
Question: What do I have? Answer: nothing
Question: What attributes does handwritten note have? Answer: indoor, readable
Question: What attributes does southwest have? Answer: indoor
Question: What attributes does large pizza box have? Answer: indoor
```

Seq2Seq

For both tasks, models were trained until validation accuracy (picked to be a random 10% subset of the training data) did not improve for 5 epochs or 72 wall clock hours on a machine with 4 Nvidia GeForce RTX 2080 GPUs, three times with three random seeds. All models decode using beam search with a beam width of 15 at test time until the end-of-sequence tag is reached. The size of the decoding vocabulary for the action prediction task is 11056 and for the graph prediction task is 6985. Hyperparameters were not tuned and were taken from BART (Lewis *et al.* 2020).

Table A.1: Hyperparameters used to train the Seq2Seq model. It has a total of 232 million trainable parameters.

Hyperparameter type	**Value**
Dictionary Tokenizer	Byte-pair encoding
Num. Encoder layers	6
Num. Decoder layers	6
Num. encoder and decoder attention heads	8
Feedforward network hidden size	4096
Input length	1024
Embedding size	768
Batch size	16
Dropout ratio	0.1
Gradient clip	1.0
Optimizer	Adam
Learning rate	10×10^{-4}

A.3 Worldformer

All baseline models have hyperparameters taken from their respective works and from the JerichoWorld benchmarks. They are trained accordingly, with the exception of GATA-World. This model uses an architecture identical to that of the Worldformer but is trained

Table A.2: Hyperparameters used to train the Worldformer. It has a total of ≈ 380 million trainable parameters. The triple tokenizer splits on individual parts of $\langle s, r, o \rangle$.

Hyperparameter type	Value
Text encoder	
Dictionary Tokenizer	Sentence piece
Num. layers	6
Num. attention heads	6
Feedforward network hidden size	3072
Input length	1024
Embedding size	768
Graph encoder	
Dictionary Tokenizer	Triple tokenizer
Num. layers	6
Num. attention heads	6
Feedforward network hidden size	3072
Input length	1024
Embedding size	768
Aggregator	
Num. layers	2
Num. attention heads	2
Feedforward network hidden size	4096
Input length	2048
Embedding size	768
Action Decoder	
Dictionary Tokenizer	White space tokenizer
Num. layers	6
Num. attention heads	6
Feedforward network hidden size	3072
Input length	1024
Embedding size	768
Graph Decoder	
Dictionary Tokenizer	Triple tokenizer
Num. layers	6
Num. attention heads	6
Feedforward network hidden size	3072
Input length	1024
Embedding size	768
Common	
Activation	gelu
Batch size	16
Dropout ratio	0.1
Gradient clip	1.0
Optimizer	Adam
Learning rate	3×10^{-4}

to predict add/del rules as described earlier—i.e. it is trained single-task and has only a graph decoder and no action decoder. The hyperparameters and training methodology for this model match those of the Worldformer described below.

Following JerichoWorld models were trained until validation accuracy (picked to be a random 10% subset of the training data) did not improve for 5 epochs or 96 wall clock hours on a machine with 4 Nvidia GeForce RTX 2080 GPUs, three times with three random seeds. All models decode using beam search with a beam width of 15 at test time until the end-of-sequence tag is reached. The size of the decoding vocabulary for the action decoder

is 11056 and for the graph decoder is 7002. Hyperparameters were not tuned and were taken from other transformer-based text game works (Adhikari *et al.* 2020; Ammanabrolu *et al.* 2020d). Hyperparameter settings for ablations do not vary from the full Worldformer.

Encoders have an architecture similar to BERT (Devlin *et al.* 2019) and decoders one similar to GPT-2 (Radford *et al.* 2019)—the rest of the hyperparameters are provided in Table A.2.

A.3.1 Example Output Graphs and Actions

Here, we provide 3 examples of graphs and actions generated from the randomly drawn test example instances shown in JerichoWorld to provide a qualitative comparison across the different models.

```
Game: ludicorp
State:
   Location: Meeting Area
          A door to the south leads into the garden. A water cooler sits invitingly in
              the corner. More doors lead east and west. You can see a Coil of wire here.
   Observation: Dropped.
   Inventory: You are carrying:
              a Dragon Statue
              some Plant Pots
              a Long Ladder
              a Gun
   Graph: ["Coil of wire", "in", "Meeting Area"],
      ["you", "have", "Plant Pots"],
      ["Water Cooler", "in", "Meeting Area"],
      ["you", "have", "Dragon Statue"],
      ["you", "have", "Long Ladder"],
      ["you", "in", "Meeting Area"],
      ["you", "have", "Gun"]
   Valid Actions: take wire, east, west, south, put dragon down, put pots down, put gun
       down, put ladder down
Act: take wire
Next State:
   Location: Meeting Area
          A door to the south leads into the garden. A water cooler sits invitingly in
              the corner. More doors lead east and west.
```

Observation: Taken.

Inventory: You are carrying:

a Coil of wire

a Dragon Statue

some Plant Pots

a Long Ladder

a Gun

Graph: ["you", "have", "Coil of wire"],

["you", "have", "Plant Pots"],

["Water Cooler", "in", "Meeting Area"],

["you", "have", "Dragon Statue"],

["you", "have", "Long Ladder"],

["you", "in", "Meeting Area"],

["you", "have", "Gun"]

Valid Actions: put wire down, east, west, south, put dragon down, put pots down, put gun down, put ladder down

Predicted Next State Graphs:

Rules: ["door to the", "in", "South"],

["more doors", "in", "east and west"]

["leads to the", "in", "garden"],

["you are carrying", "have", "some Plant Pots"],

["a water cooler sits", "in", "corner"],

["you are carrying", "have", "Dragon Statue"],

["you are carrying", "have", "a Long Ladder"],

["you", "in", "Meeting Area"],

["you are carrying", "have", "a Gun"]

QA: ["you", "have", "Coil of wire"],

["door", "in", "South"],

["doors", "in", "east and west"]

["you", "have", "some Plant Pots"],

["water cooler", "in", "corner"],

["you", "have", "Dragon Statue"],

["you", "have", "a Long Ladder a Gun"],

["you", "in", "Meeting Area"]

Seq2Seq: ["you", "have", "Pots"],

["you", "in", "Statue"],

["you", "have", "Ladder Gun"],

["you", "in", "Meeting Area"]

GATA-W: ["you", "in", "Coil of wire"],

["you", "have", "Plant Pots"],

["Water Cooler", "in", "Meeting Area"],

["you", "have", "Dragon Statue"],
["you", "in", "Statue"],
["you", "have", "Long Ladder"],
["you", "in", "Meeting Area"],
["you", "have", "Gun"]
Worldformer: ["you", "have", "Coil of wire"],
["you", "have", "Plant Pots"],
["Water Cooler", "in", "Meeting Area"],
["you", "in", "Dragon Statue"],
["you", "have", "Long Ladder"],
["you", "in", "Meeting Area"],
["you", "have", "Gun"]
Predicted Next State Valid Actions:
Seq2Seq Actions: east, west, south, north, pick coil up, pick statue up, put ladder gun down, put meeting area down, put pots down
CALM Actions: northwest, up, take plant, put plant in cooler, take water, take fish, south, wait, take all, north, take dragon, take cooler, southeast, take dragon statue, get all, east, west, take gun, northeast, southwest
Worldformer Actions: east, west, south, north, put coil down, pick statue up, put ladder down, put pots down, put gun down

==========

Game: pentari
State:
Location: Armory
Many death-dealing weapons of every type were stored here. Several tall racks probably held spears while shorter ones mounted against the wall stored various kinds of swords. Other wall mounts, also empty, give you no idea what sort of weapons may have been held by them. A large archway north is partially blocked by collapsed stones and rubble. You can see a jewel encrusted dagger here.
Observation: Armory
Many death-dealing weapons of every type were stored here. Several tall racks probably held spears while shorter ones mounted against the wall stored various kinds of swords. Other wall mounts, also empty, give you no idea what sort of weapons may have been held by them. A large archway north is partially blocked by collapsed stones and rubble. You can see a jewel encrusted dagger here.
Inventory: You are carrying nothing.
Graph: ["jewel encrusted dagger", "in", "Armory"],

["Armory", "west", "Main Hall"],

["you", "in", "Armory"]

Valid Actions: take other, east, north

Act: east

Next State:

Location: Main Hall

This once majestic room was where visitors would come to relax and meet with the formal lord of the castle in a somewhat informal atmosphere. Several large comfortable couches are scattered about, dusty and altogether squalid . Many large tapestries still hang on the walls but are horribly faded from age. Large open archways lead east and west while a huge fireplace dominates the center of the room against the northern wall.

Observation: Main Hall

This once majestic room was where visitors would come to relax and meet with the formal lord of the castle in a somewhat informal atmosphere. Several large comfortable couches are scattered about, dusty and altogether squalid. Many large tapestries still hang on the walls but are horribly faded from age. Large open archways lead east and west while a huge fireplace dominates the center of the room against the northern wall.

Inventory: You are carrying nothing.

Graph: ["couch", "in", "Main Hall"],

["jewel encrusted dagger", "in", "Armory"],

["you", "in", "Main Hall"],

["Main Hall", "east", "Armory"],

["tapestry", "in","Main Hall"]

Valid Actions: east, west, south, north

Predicted Next State Graphs:

Rules: ["Several large comfortable couches", "in", "Main Hall"],

["many large tapestries", "in", "Main Hall"],

["many large tapestries", "is", "age"],

["huge fireplace dominates", "in", "center of room against northern wall"],

["death dealing weapons", "in", "Armory"],

["visitors", "in", "to relax"],

["archway north", "is", "blocked"],

["archway north", "in", "collapsed stones"],

["spears", "in", "several tall racks"],

["you", "in", "Armory"],

["you", "in", "see encrusted dagger"],

["you are carrying", "have", "nothing"],

["Main Hall", "east", "Armory"]

QA: ["couches", "in", "Main Hall"],

["tapestries", "in", "Main Hall"],
["tapestries", "is", "faded"],
["Main Hall", "east", "Armory"],
["fireplace", "in", "center of room"],
["weapons", "in", "Armory"],
["visitors", "in", "Armory"],
["spears", "is", "tall"],
["you", "in", "Armory"],
["you", "have", "nothing"]

Seq2Seq: ["couch", "in", "Main Hall"],
["jewel", "in", "Armory"],
["you", "in", "Main Hall"],
["large", "in","Main Hall"]

GATA-W: ["couch", "in", "Main Hall"],
["jewel encrusted dagger", "in", "Armory"],
["you", "in", "Main Hall"],
["tapestry", "in","Main Hall"]

Worldformer: ["couch", "in", "Main Hall"],
["dagger", "in", "Armory"],
["you", "in", "Main Hall"],
["Main Hall", "east", "Armory"],
["faded", "in", "Main Hall"]

Predicted Next State Valid Actions:

Seq2Seq Actions: east, west, south, north, take jewel, take large, take armory, put couch down

CALM Actions: get dagger, northwest, out, up, down, exits, search tapestries, close door, south, search fireplace, enter fireplace, open fireplace, north, in, southeast, east, west, take dagger, northeast, southwest

Worldformer Actions: east, west, south, north, take jewel, take large, examine large, examine couch

==========

Game: temple

State:

Location: Dead End

This part of the town is radically different from the parts closer to the tower. The roads are narrower and the paving is irregular, sometimes stone slabs and sometimes cobblestones. The buildings are tall but less well kept than before. There are still no windows or doors, but there are a few overhead bridges from house to house. The road ends here and the only way out is to

the north. You can also see a wrought iron key and Charles Bristow here.

Observation: The cat jumps aside to avoid the projectile, but moves a bit too far. It falls down, but like most cats it escapes unhurt. The cat runs off to the north. The wrought iron key falls down again and hits one of the stone slabs. There is a hollow sound, much like there was some cavity below the slab. 'I used to have a cat , you know' Charles remarks.

Inventory: You are carrying:

a vial labelled Mukhtar

the Caelestae Horriblis

two vials

a yellow paper

a hideous statue

Graph: ["stone slab", "in", "Dead End"],

["slab", "is", "animate"],

["Charles' clothes", "in", "Charles Bristow"],

["clothes", "is", "animate"],

["clothes", "is", "equip"],

["you", "have", "mysterious vial"],

["you", "have", "vial labelled Mukhtar"],

["overhead bridge", "in", "Dead End"],

["you", "have", "Caelestae Horriblis"],

["you", "have", "yellow paper"],

["yellow paper", "is", "animate"],

["you", "have", "hideous statue"],

["hideous statue", "is", "animate"],

["sky", "in", "Dead End"],

["elliptcal building", "in", "Dead End"],

["you", "in", "Dead End"],

["entrance", "in", "Dead End"],

["Charles Bristow", "in", "Dead End"],

["wrought iron key", "in", "Dead End"]

Valid Actions: take wrought, take paving, north, put mysterious down, put caelestae down, put paper down, put statue down, put mukhtar down, drop wrought against bridge

Act: put paper down

Next State:

Location: Dead End

This part of the town is radically different from the parts closer to the tower . The roads are narrower and the paving is irregular, sometimes stone slabs and sometimes cobblestones. The buildings are tall but less well kept than before. There are still no windows or doors, but there are a few overhead

bridges from house to house. The road ends here and the only way out is to the north. You can also see a yellow paper, a wrought iron key and Charles Bristow here.

Observation: Dropped.

Inventory: You are carrying:

a vial labelled Mukhtar

the Caelestae Horriblis

two vials

a hideous statue

Graph: ["yellow paper", "in", "Dead End"],

["yellow paper", "is", "animate"],

["stone slab", "in", "Dead End"],

["stone slab", "is", "animate"],

["Charles' clothes", "in", "Charles Bristow"],

["clothes", "is", "animate"],

["clothes", "is", "equip"],

["you", "have", "mysterious vial"],

["you", "have", "vial labelled Mukhtar"],

["overhead bridge", "in", "Dead End"],

["you", "have", "Caelestae Horriblis"],

["you", "have", "hideous statue"],

["hideous statue", "is", "animate"],

["sky", "in", "Dead End"],

["elliptcal building", "in", "Dead End"],

["you", "in", "Dead End"],

["entrance", "in", "Dead End"],

["Charles Bristow", "in", "Dead End"],

["wrought iron key", "in", "Dead End"]

Valid Actions: take wrought, take paving, north, put mysterious down, put caelestae down, take all, put statue down, put mukhtar down, drop wrought against bridge

Predicted Next State Graphs:

Rules:["you", "in", "Dead End"],

["Dead End", "is", "radically different from the parts closer to the tower"],

["roads", "is", "narrower"],

["the paving", "is", "irregular"],

["the buildings", "is", "less well kept"],

["only way", "is", "north"],

["you can see", "in", "yellow paper"],

["you can see", "in", "a wrought iron key and Charles Bristow here"],

["you are carrying", "have", "a vial labelled Mukhtar"],

["you are carrying", "have", "the Caelestae Horriblis two vials"],

["you are carrying", "have", "a hideous statue"]

QA: ["you", "in", "Dead End"],
["Dead End", "is", "animate"],
["road", "is", "animate"],
["paving", "is", "animate"],
["building", "is", "animate"],
["yellow paper", "in", "Dead End"],
["paper", "is", "animate"],
["key", "in", "Dead End"],
["key", "is", "animate"],
["Charles Bristow", "in", "Dead End"],
["Charles Bristow", "is", "animate"],
["you", "have", "a vial"],
["a vial", "is", "animate"],
["you", "have", "Caelestae Horriblis"],
["Caelestae Horriblis", "is", "animate"],
["you", "have", "two vials"],
["you", "have", "a hideous statue"]
["a hideous statue", "is", "animate"]

Seq2Seq: ["you", "in", "Dead End"],
["Dead End", "is", "animate"],
["key", "in", "Dead End"],
["key", "is", "animate"],
["Charles Bristow", "in", "Dead End"],
["Charles Bristow", "is", "animate"],
["you", "have", "vial"],
["you", "have", "two vials"],
["you", "in", "statue"]

GATA-W: ["yellow paper", "in", "Dead End"],
["stone slab", "in", "Dead End"],
["stone slab", "is", "animate"],
["Charles' clothes", "in", "Charles Bristow"],
["clothes", "is", "animate"],
["clothes", "is", "equip"],
["you", "have", "mysterious vial"],
["you", "have", "vial labelled Mukhtar"],
["overhead bridge", "in", "Dead End"],
["you", "in", "Caelestae Horriblis"],
["sky", "in", "Dead End"],
["elliptcal building", "in", "Dead End"],
["you", "in", "Dead End"],

```
      ["entrance", "in", "Dead End"],
      ["Charles Bristow", "in", "Dead End"],
      ["wrought iron key", "in", "Dead End"]
   Worldformer: ["yellow paper", "in", "Dead End"],
      ["yellow paper", "is", "animate"],
      ["stone slab", "in", "Dead End"],
      ["stone slab", "is", "animate"],
      ["Charles' clothes", "in", "Charles Bristow"],
      ["clothes", "is", "animate"],
      ["clothes", "is", "equip"],
      ["you", "have", "vial"],
      ["bridge", "in", "Dead End"],
      ["you", "have", "Caelestae Horriblis"],
      ["you", "have", "hideous statue"],
      ["hideous statue", "is", "animate"],
      ["sky", "in", "Dead End"],
      ["building", "in", "Dead End"],
      ["you", "in", "Dead End"],
      ["entrance", "in", "Dead End"],
      ["Charles Bristow", "in", "Dead End"],
      ["wrought iron key", "in", "Dead End"]
Predicted Next State Valid Actions:
   Seq2Seq Actions: east, west, south, north, drop vial, take key, take statue, take
       charles bristow
   CALM Actions: put paper down, drop vial, take key, up, down, put paper in cavity, put
       yellow paper down, take vial, drop key, drop all, open vial, south, get key, put
       yellow paper in cavity, take all, put vial in cavity, north, east, west, give vial
       to cat
   Worldformer Actions: south, east, west, north, take building, take entrance, drop vial,
        take charles bristow, take paper, drop vials, take sky, drop key, take statue,
       take bridge
```

Here the first 50 steps of the transcript of an agent following a random valid action policy in *Zork1* is provided along with the predicted knowledge graph at each step as a qualitative measure of compounding error of such models.

```
==========STEP 1==========
Observation:
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 here.
You are emptyhanded.
```

Copyright (c) 1981, 1982, 1983 Infocom, Inc. All rights reserved. ORK is a registered trademark of Infocom, Inc. Revision 88 / Serial number 8402 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 here.

look

Action: north

[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', 'in', 'west'), ('mailbox', 'is', 'animate'), ('mailbox', 'in', 'west'), ('you', 'is', 'you'), ('you', 'in', 'west')]

==========STEP 2==========

Observation:

North of House You are facing the north side of a white house. There is no door here, and all the windows are boarded up. To the north a narrow path winds through the trees.

You are emptyhanded.

North of House You are facing the north side of a white house. There is no door here, and all the windows are boarded up. To the north a narrow path winds through the trees.

north

Action: west

[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', 'in', 'west'), ('mailbox', 'is', 'animate'), ('mailbox', 'in', 'west'), ('you', 'is', 'you'), ('you', 'in', 'north_of_house'), ('north_of_house', 'north', 'west'), ('windows', 'in', 'north_of_house'), ('windows', 'is', 'animate'), ('trees', 'in', 'north_of_house'), ('trees', 'is', 'animate'), ('path', 'is', 'animate'), ('path', 'in', 'north_of_house'), ('north', 'is', 'animate'), ('north', 'in', 'north_of_house')]

==========STEP 3==========

Observation:

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 here.

You are emptyhanded.

West of House There is a small mailbox here.

west

Action: north

[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', 'in', 'west'), ('west', 'west', 'north_of_house'), ('mailbox', 'is', 'animate'), ('mailbox', 'in', 'west'), ('you', 'is', 'you'), ('you', 'in', 'west'), ('north_of_house', 'north', 'west'), ('windows', 'in', 'north_of_house'), ('windows', 'is', 'animate'), ('trees', 'in', 'north_of_house'), ('trees', 'is', 'animate'), ('path', 'is', 'animate'), ('path', 'in', 'north_of_house'), ('north', 'is', 'animate'), ('north', 'in

', 'north_of_house')]

==========STEP 4==========

Observation:

North of House You are facing the north side of a white house. There is no door here, and all the windows are boarded up. To the north a narrow path winds through the trees.

You are emptyhanded.

North of House

north

Action: north

[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', 'in', 'west'), ('west', 'west', 'north_of_house'), ('mailbox', 'is', 'animate'), ('mailbox', 'in', 'west'), ('you', 'is', 'you'), ('you', 'in', 'north_of_house'), ('north_of_house', 'north', 'west'), ('windows', 'in', 'north_of_house'), ('windows', 'is', 'animate'), ('trees', 'in', 'north_of_house'), ('trees', 'is', 'animate'), ('path', 'is', 'animate'), ('path', 'in', 'north_of_house'), ('north', 'is', 'animate'), ('north', 'in', 'north_of_house')]

==========STEP 5==========

Observation:

Forest Path This is a path winding through a dimly lit forest. The path heads northsouth here. One particularly large tree with some low branches stands at the edge of the path.

You are emptyhanded.

Forest Path This is a path winding through a dimly lit forest. The path heads northsouth here. One particularly large tree with some low branches stands at the edge of the path. You hear in the distance the chirping of a song bird.

north

Action: east

[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', 'in', 'west'), ('west', 'west', 'north_of_house'), ('mailbox', 'is', 'animate'), ('mailbox', 'in', 'west'), ('you', 'is', 'you'), ('you', 'in', 'forest_path'), ('north_of_house', 'north', 'west'), ('windows', 'in', 'north_of_house'), ('windows', 'is', 'animate'), ('trees', 'in', 'north_of_house'), ('trees', 'is', 'animate'), ('path', 'is', 'animate'), ('path', 'in', 'north_of_house'), ('north', 'is', 'animate'), ('north', 'in', 'north_of_house'), ('large', 'in', 'forest_path'), ('large', 'is', 'animate'), ('forest_path', 'north', 'north_of_house'), ('tree', 'in', 'forest_path'), ('tree', 'is', 'animate'), ('south', 'is', 'animate'), ('south', 'in', 'forest_path')]

==========STEP 6==========

Observation:

Forest This is a dimly lit forest, with large trees all around.

```
You are emptyhanded.
Forest This is a dimly lit forest, with large trees all around.
east

Action: east
[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', '
    in', 'west'), ('west', 'west', 'north_of_house'), ('mailbox', 'is', 'animate'), ('
    mailbox', 'in', 'west'), ('you', 'is', 'you'), ('you', 'in', 'forest'), ('
    north_of_house', 'north', 'west'), ('windows', 'in', 'north_of_house'), ('windows', '
    is', 'animate'), ('trees', 'in', 'north_of_house'), ('trees', 'is', 'animate'), ('
    trees', 'in', 'forest'), ('path', 'is', 'animate'), ('path', 'in', 'north_of_house'),
    ('north', 'is', 'animate'), ('north', 'in', 'north_of_house'), ('large', 'in', '
    forest_path'), ('large', 'is', 'animate'), ('large', 'in', 'forest'), ('forest_path',
    'north', 'north_of_house'), ('tree', 'in', 'forest_path'), ('tree', 'is', 'animate'),
    ('south', 'is', 'animate'), ('south', 'in', 'forest_path'), ('forest', 'is', 'animate
    '), ('forest', 'east', 'forest_path'), ('forest', 'in', 'forest')]
==========STEP 7==========
Observation:
Forest The forest thins out, revealing impassable mountains.
You are emptyhanded.
Forest The forest thins out, revealing impassable mountains.
east

Action: up
[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', '
    in', 'west'), ('west', 'west', 'north_of_house'), ('mailbox', 'is', 'animate'), ('
    mailbox', 'in', 'west'), ('you', 'is', 'you'), ('north_of_house', 'north', 'west'), ('
    windows', 'in', 'north_of_house'), ('windows', 'is', 'animate'), ('trees', 'in', '
    north_of_house'), ('trees', 'is', 'animate'), ('trees', 'in', 'forest'), ('path', 'is
    ', 'animate'), ('path', 'in', 'north_of_house'), ('north', 'is', 'animate'), ('north',
     'in', 'north_of_house'), ('large', 'in', 'forest_path'), ('large', 'is', 'animate'),
    ('large', 'in', 'forest'), ('forest_path', 'north', 'north_of_house'), ('tree', 'in',
    'forest_path'), ('tree', 'is', 'animate'), ('south', 'is', 'animate'), ('south', 'in',
     'forest_path'), ('forest', 'is', 'animate'), ('forest', 'east', 'forest_path'), ('
    forest', 'in', 'forest'), ('mountains', 'is', 'animate'), ('mountains', 'in', 'forest
    ')]
==========STEP 8==========
Observation:
Forest The forest thins out, revealing impassable mountains.
You are emptyhanded.
The mountains are impassable.
```

```
up

Action: west
[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', '
    in', 'west'), ('west', 'west', 'north_of_house'), ('mailbox', 'is', 'animate'), ('
    mailbox', 'in', 'west'), ('you', 'is', 'you'), ('north_of_house', 'north', 'west'), ('
    windows', 'in', 'north_of_house'), ('windows', 'is', 'animate'), ('trees', 'in', '
    north_of_house'), ('trees', 'is', 'animate'), ('trees', 'in', 'forest'), ('path', 'is
    ', 'animate'), ('path', 'in', 'north_of_house'), ('north', 'is', 'animate'), ('north',
     'in', 'north_of_house'), ('large', 'in', 'forest_path'), ('large', 'is', 'animate'),
    ('large', 'in', 'forest'), ('forest_path', 'north', 'north_of_house'), ('tree', 'in',
    'forest_path'), ('tree', 'is', 'animate'), ('south', 'is', 'animate'), ('south', 'in',
     'forest_path'), ('forest', 'is', 'animate'), ('forest', 'east', 'forest_path'), ('
    forest', 'up', 'forest'), ('mountains', 'is', 'animate'), ('mountains', 'in', 'forest
    ')]
==========STEP 9==========
Observation:
Forest This is a dimly lit forest, with large trees all around.
You are emptyhanded.
Forest
west

Action: west
[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', '
    in', 'west'), ('west', 'west', 'north_of_house'), ('mailbox', 'is', 'animate'), ('
    mailbox', 'in', 'west'), ('you', 'is', 'you'), ('north_of_house', 'north', 'west'), ('
    windows', 'in', 'north_of_house'), ('windows', 'is', 'animate'), ('trees', 'in', '
    north_of_house'), ('trees', 'is', 'animate'), ('trees', 'in', 'forest'), ('path', 'is
    ', 'animate'), ('path', 'in', 'north_of_house'), ('north', 'is', 'animate'), ('north',
     'in', 'north_of_house'), ('large', 'in', 'forest_path'), ('large', 'is', 'animate'),
    ('large', 'in', 'forest'), ('forest_path', 'north', 'north_of_house'), ('tree', 'in',
    'forest_path'), ('tree', 'is', 'animate'), ('south', 'is', 'animate'), ('south', 'in',
     'forest_path'), ('forest', 'is', 'animate'), ('forest', 'east', 'forest_path'), ('
    forest', 'in', 'forest'), ('mountains', 'is', 'animate'), ('mountains', 'in', 'forest
    ')]
==========STEP 10==========
Observation:
Forest Path This is a path winding through a dimly lit forest. The path heads northsouth
    here. One particularly large tree with some low branches stands at the edge of the
    path.
You are emptyhanded.
```

```
Forest Path
west

Action: north
[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', '
    in', 'west'), ('west', 'west', 'north_of_house'), ('mailbox', 'is', 'animate'), ('
    mailbox', 'in', 'west'), ('you', 'is', 'you'), ('you', 'in', 'forest_path'), ('
    north_of_house', 'north', 'west'), ('windows', 'in', 'north_of_house'), ('windows', '
    is', 'animate'), ('trees', 'in', 'north_of_house'), ('trees', 'is', 'animate'), ('
    trees', 'in', 'forest'), ('path', 'is', 'animate'), ('path', 'in', 'north_of_house'),
    ('north', 'is', 'animate'), ('north', 'in', 'north_of_house'), ('large', 'in', '
    forest_path'), ('large', 'is', 'animate'), ('large', 'in', 'forest'), ('forest_path',
    'north', 'north_of_house'), ('forest_path', 'west', 'forest'), ('tree', 'in', '
    forest_path'), ('tree', 'is', 'animate'), ('south', 'is', 'animate'), ('south', 'in',
    'forest_path'), ('forest', 'is', 'animate'), ('forest', 'east', 'forest_path'), ('
    forest', 'in', 'forest'), ('mountains', 'is', 'animate'), ('mountains', 'in', 'forest
    ')]
==========STEP 11==========
Observation:
Clearing You are in a clearing, with a forest surrounding you on all sides. A path leads
    south. On the ground is a pile of leaves.
You are emptyhanded.
Clearing You are in a clearing, with a forest surrounding you on all sides. A path leads
    south. On the ground is a pile of leaves.
north

Action: south
[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', '
    in', 'west'), ('west', 'west', 'north_of_house'), ('mailbox', 'is', 'animate'), ('
    mailbox', 'in', 'west'), ('you', 'is', 'you'), ('you', 'in', 'clearing'), ('
    north_of_house', 'north', 'west'), ('windows', 'in', 'north_of_house'), ('windows', '
    is', 'animate'), ('trees', 'in', 'north_of_house'), ('trees', 'is', 'animate'), ('
    trees', 'in', 'forest'), ('path', 'is', 'animate'), ('path', 'in', 'north_of_house'),
    ('path', 'in', 'clearing'), ('north', 'is', 'animate'), ('north', 'in', '
    north_of_house'), ('large', 'in', 'forest_path'), ('large', 'is', 'animate'), ('large
    ', 'in', 'forest'), ('forest_path', 'north', 'north_of_house'), ('forest_path', 'west
    ', 'forest'), ('tree', 'in', 'forest_path'), ('tree', 'is', 'animate'), ('south', 'is
    ', 'animate'), ('south', 'in', 'forest_path'), ('forest', 'is', 'animate'), ('forest',
     'east', 'forest_path'), ('forest', 'in', 'forest'), ('forest', 'in', 'clearing'), ('
    mountains', 'is', 'animate'), ('mountains', 'in', 'forest'), ('clearing', 'north', '
    forest_path'), ('pile', 'in', 'clearing'), ('pile', 'is', 'animate')]
```

==========STEP 12==========

Observation:

Forest Path This is a path winding through a dimly lit forest. The path heads northsouth here. One particularly large tree with some low branches stands at the edge of the path.

You are emptyhanded.

Forest Path

south

Action: south

[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', 'in', 'west'), ('west', 'west', 'north_of_house'), ('mailbox', 'is', 'animate'), ('mailbox', 'in', 'west'), ('you', 'is', 'you'), ('you', 'in', 'forest_path'), ('north_of_house', 'north', 'west'), ('windows', 'in', 'north_of_house'), ('windows', 'is', 'animate'), ('trees', 'in', 'north_of_house'), ('trees', 'is', 'animate'), ('trees', 'in', 'forest'), ('path', 'is', 'animate'), ('path', 'in', 'north_of_house'), ('path', 'in', 'clearing'), ('north', 'is', 'animate'), ('north', 'in', 'north_of_house'), ('large', 'in', 'forest_path'), ('large', 'is', 'animate'), ('large', 'in', 'forest'), ('forest_path', 'north', 'north_of_house'), ('forest_path', 'west', 'forest'), ('forest_path', 'south', 'clearing'), ('tree', 'in', 'forest_path'), ('tree', 'is', 'animate'), ('south', 'is', 'animate'), ('south', 'in', 'forest_path'), ('forest', 'is', 'animate'), ('forest', 'east', 'forest_path'), ('forest', 'in', 'forest'), ('forest', 'in', 'clearing'), ('mountains', 'is', 'animate'), ('mountains', 'in', 'forest'), ('clearing', 'north', 'forest_path'), ('pile', 'in', 'clearing'), ('pile', 'is', 'animate')]

==========STEP 13==========

Observation:

North of House You are facing the north side of a white house. There is no door here, and all the windows are boarded up. To the north a narrow path winds through the trees.

You are emptyhanded.

North of House

south

Action: north

[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', 'in', 'west'), ('west', 'west', 'north_of_house'), ('mailbox', 'is', 'animate'), ('mailbox', 'in', 'west'), ('you', 'is', 'you'), ('you', 'in', 'north_of_house'), ('north_of_house', 'north', 'west'), ('north_of_house', 'south', 'forest_path'), ('windows', 'in', 'north_of_house'), ('windows', 'is', 'animate'), ('trees', 'in', 'north_of_house'), ('trees', 'is', 'animate'), ('trees', 'in', 'forest'), ('path', 'is', 'animate'), ('path', 'in', 'north_of_house'), ('path', 'in', 'clearing'), ('north',

'is', 'animate'), ('north', 'in', 'north_of_house'), ('large', 'in', 'forest_path'), ('large', 'is', 'animate'), ('large', 'in', 'forest'), ('forest_path', 'north', 'north_of_house'), ('forest_path', 'west', 'forest'), ('forest_path', 'south', 'clearing'), ('tree', 'in', 'forest_path'), ('tree', 'is', 'animate'), ('south', 'is', 'animate'), ('south', 'in', 'forest_path'), ('forest', 'is', 'animate'), ('forest', 'east', 'forest_path'), ('forest', 'in', 'forest'), ('forest', 'in', 'clearing'), ('mountains', 'is', 'animate'), ('mountains', 'in', 'forest'), ('clearing', 'north', 'forest_path'), ('pile', 'in', 'clearing'), ('pile', 'is', 'animate')]

==========STEP 14==========

Observation:

Forest Path This is a path winding through a dimly lit forest. The path heads northsouth here. One particularly large tree with some low branches stands at the edge of the path.

You are emptyhanded.

Forest Path

north

Action: north

[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', 'in', 'west'), ('west', 'west', 'north_of_house'), ('mailbox', 'is', 'animate'), ('mailbox', 'in', 'west'), ('you', 'is', 'you'), ('you', 'in', 'forest_path'), ('north_of_house', 'north', 'west'), ('north_of_house', 'south', 'forest_path'), ('windows', 'in', 'north_of_house'), ('windows', 'is', 'animate'), ('trees', 'in', 'north_of_house'), ('trees', 'is', 'animate'), ('trees', 'in', 'forest'), ('path', 'is', 'animate'), ('path', 'in', 'north_of_house'), ('path', 'in', 'clearing'), ('north', 'is', 'animate'), ('north', 'in', 'north_of_house'), ('large', 'in', 'forest_path'), ('large', 'is', 'animate'), ('large', 'in', 'forest'), ('forest_path', 'north', 'north_of_house'), ('forest_path', 'west', 'forest'), ('forest_path', 'south', 'clearing'), ('tree', 'in', 'forest_path'), ('tree', 'is', 'animate'), ('south', 'is', 'animate'), ('south', 'in', 'forest_path'), ('forest', 'is', 'animate'), ('forest', 'east', 'forest_path'), ('forest', 'in', 'forest'), ('forest', 'in', 'clearing'), ('mountains', 'is', 'animate'), ('mountains', 'in', 'forest'), ('clearing', 'north', 'forest_path'), ('pile', 'in', 'clearing'), ('pile', 'is', 'animate')]

==========STEP 15==========

Observation:

Clearing You are in a clearing, with a forest surrounding you on all sides. A path leads south. On the ground is a pile of leaves.

You are emptyhanded.

Clearing On the ground is a pile of leaves.

north

```
Action: west
[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', '
    in', 'west'), ('west', 'west', 'north_of_house'), ('mailbox', 'is', 'animate'), ('
    mailbox', 'in', 'west'), ('you', 'is', 'you'), ('you', 'in', 'clearing'), ('
    north_of_house', 'north', 'west'), ('north_of_house', 'south', 'forest_path'), ('
    windows', 'in', 'north_of_house'), ('windows', 'is', 'animate'), ('trees', 'in', '
    north_of_house'), ('trees', 'is', 'animate'), ('trees', 'in', 'forest'), ('path', 'is
    ', 'animate'), ('path', 'in', 'north_of_house'), ('path', 'in', 'clearing'), ('north',
     'is', 'animate'), ('north', 'in', 'north_of_house'), ('large', 'in', 'forest_path'),
    ('large', 'is', 'animate'), ('large', 'in', 'forest'), ('forest_path', 'north', '
    north_of_house'), ('forest_path', 'west', 'forest'), ('forest_path', 'south', '
    clearing'), ('tree', 'in', 'forest_path'), ('tree', 'is', 'animate'), ('south', 'is',
    'animate'), ('south', 'in', 'forest_path'), ('forest', 'is', 'animate'), ('forest', '
    east', 'forest_path'), ('forest', 'in', 'forest'), ('forest', 'in', 'clearing'), ('
    mountains', 'is', 'animate'), ('mountains', 'in', 'forest'), ('clearing', 'north', '
    forest_path'), ('pile', 'in', 'clearing'), ('pile', 'is', 'animate'), ('leaves', 'is',
     'animate'), ('leaves', 'in', 'clearing')]
==========STEP 16==========
Observation:
Forest This is a forest, with trees in all directions. To the east, there appears to be
    sunlight.
You are emptyhanded.
Forest This is a forest, with trees in all directions. To the east, there appears to be
    sunlight.
west

Action: go around forest
[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', '
    in', 'west'), ('west', 'west', 'north_of_house'), ('mailbox', 'is', 'animate'), ('
    mailbox', 'in', 'west'), ('you', 'is', 'you'), ('you', 'in', 'forest'), ('
    north_of_house', 'north', 'west'), ('north_of_house', 'south', 'forest_path'), ('
    windows', 'in', 'north_of_house'), ('windows', 'is', 'animate'), ('trees', 'in', '
    north_of_house'), ('trees', 'is', 'animate'), ('trees', 'in', 'forest'), ('path', 'is
    ', 'animate'), ('path', 'in', 'north_of_house'), ('path', 'in', 'clearing'), ('north',
     'is', 'animate'), ('north', 'in', 'north_of_house'), ('large', 'in', 'forest_path'),
    ('large', 'is', 'animate'), ('large', 'in', 'forest'), ('forest_path', 'north', '
    north_of_house'), ('forest_path', 'west', 'forest'), ('forest_path', 'south', '
    clearing'), ('tree', 'in', 'forest_path'), ('tree', 'is', 'animate'), ('south', 'is',
    'animate'), ('south', 'in', 'forest_path'), ('forest', 'is', 'animate'), ('forest', '
    east', 'forest_path'), ('forest', 'in', 'forest'), ('forest', 'west', 'clearing'), ('
    mountains', 'is', 'animate'), ('mountains', 'in', 'forest'), ('clearing', 'north', '
```

forest_path'), ('pile', 'in', 'clearing'), ('pile', 'is', 'animate'), ('leaves', 'is', 'animate'), ('leaves', 'in', 'clearing'), ('east', 'is', 'animate'), ('east', 'in', 'forest')]

==========STEP 17==========

Observation:

Forest This is a dimly lit forest, with large trees all around.

You are emptyhanded.

Forest

go around forest

Action: west

[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', 'in', 'west'), ('west', 'west', 'north_of_house'), ('mailbox', 'is', 'animate'), ('mailbox', 'in', 'west'), ('you', 'is', 'you'), ('north_of_house', 'north', 'west'), ('north_of_house', 'south', 'forest_path'), ('windows', 'in', 'north_of_house'), ('windows', 'is', 'animate'), ('trees', 'in', 'north_of_house'), ('trees', 'is', 'animate'), ('trees', 'in', 'forest'), ('path', 'is', 'animate'), ('path', 'in', 'north_of_house'), ('path', 'in', 'clearing'), ('north', 'is', 'animate'), ('north', 'in', 'north_of_house'), ('large', 'in', 'forest_path'), ('large', 'is', 'animate'), ('large', 'in', 'forest'), ('forest_path', 'north', 'north_of_house'), ('forest_path', 'west', 'forest'), ('forest_path', 'south', 'clearing'), ('tree', 'in', 'forest_path'), ('tree', 'is', 'animate'), ('south', 'is', 'animate'), ('south', 'in', 'forest_path'), ('forest', 'is', 'animate'), ('forest', 'east', 'forest_path'), ('forest', 'in', 'forest'), ('forest', 'west', 'clearing'), ('mountains', 'is', 'animate'), ('mountains', 'in', 'forest'), ('clearing', 'north', 'forest_path'), ('pile', 'in', 'clearing'), ('pile', 'is', 'animate'), ('leaves', 'is', 'animate'), ('leaves', 'in', 'clearing'), ('east', 'is', 'animate'), ('east', 'in', 'forest')]

==========STEP 18==========

Observation:

Forest Path This is a path winding through a dimly lit forest. The path heads northsouth here. One particularly large tree with some low branches stands at the edge of the path.

You are emptyhanded.

Forest Path

west

Action: up

[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', 'in', 'west'), ('west', 'west', 'north_of_house'), ('mailbox', 'is', 'animate'), ('mailbox', 'in', 'west'), ('you', 'is', 'you'), ('you', 'in', 'forest_path'), ('north_of_house', 'north', 'west'), ('north_of_house', 'south', 'forest_path'), ('

windows', 'in', 'north_of_house'), ('windows', 'is', 'animate'), ('trees', 'in', '
north_of_house'), ('trees', 'is', 'animate'), ('trees', 'in', 'forest'), ('path', 'is
', 'animate'), ('path', 'in', 'north_of_house'), ('path', 'in', 'clearing'), ('north',
 'is', 'animate'), ('north', 'in', 'north_of_house'), ('large', 'in', 'forest_path'),
('large', 'is', 'animate'), ('large', 'in', 'forest'), ('forest_path', 'north', '
north_of_house'), ('forest_path', 'west', 'forest'), ('forest_path', 'south', '
clearing'), ('tree', 'in', 'forest_path'), ('tree', 'is', 'animate'), ('south', 'is',
'animate'), ('south', 'in', 'forest_path'), ('forest', 'is', 'animate'), ('forest', '
east', 'forest_path'), ('forest', 'in', 'forest'), ('forest', 'west', 'clearing'), ('
mountains', 'is', 'animate'), ('mountains', 'in', 'forest'), ('clearing', 'north', '
forest_path'), ('pile', 'in', 'clearing'), ('pile', 'is', 'animate'), ('leaves', 'is',
 'animate'), ('leaves', 'in', 'clearing'), ('east', 'is', 'animate'), ('east', 'in', '
forest')]

==========STEP 19==========

Observation:

Up a Tree You are about 10 feet above the ground nestled among some large branches. The
nearest branch above you is above your reach. Beside you on the branch is a small
birds nest. In the birds nest is a large egg encrusted with precious jewels,
apparently scavenged by a childless songbird. The egg is covered with fine gold inlay,
 and ornamented in lapis lazuli and motherofpearl. Unlike most eggs, this one is
hinged and closed with a delicate looking clasp. The egg appears extremely fragile.

You are emptyhanded.

Up a Tree You are about 10 feet above the ground nestled among some large branches. The
nearest branch above you is above your reach. Beside you on the branch is a small
birds nest. In the birds nest is a large egg encrusted with precious jewels,
apparently scavenged by a childless songbird. The egg is covered with fine gold inlay,
 and ornamented in lapis lazuli and motherofpearl. Unlike most eggs, this one is
hinged and closed with a delicate looking clasp. The egg appears extremely fragile.

up

Action: take egg

[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', '
in', 'west'), ('west', 'west', 'north_of_house'), ('mailbox', 'is', 'animate'), ('
mailbox', 'in', 'west'), ('you', 'is', 'you'), ('north_of_house', 'north', 'west'), ('
north_of_house', 'south', 'forest_path'), ('windows', 'in', 'north_of_house'), ('
windows', 'is', 'animate'), ('trees', 'in', 'north_of_house'), ('trees', 'is', '
animate'), ('trees', 'in', 'forest'), ('path', 'is', 'animate'), ('path', 'in', '
north_of_house'), ('path', 'in', 'clearing'), ('north', 'is', 'animate'), ('north', '
in', 'north_of_house'), ('large', 'in', 'forest_path'), ('large', 'is', 'animate'), ('
large', 'in', 'forest'), ('forest_path', 'north', 'north_of_house'), ('forest_path', '
west', 'forest'), ('forest_path', 'south', 'clearing'), ('tree', 'in', 'forest_path'),

('tree', 'is', 'animate'), ('south', 'is', 'animate'), ('south', 'in', 'forest_path'), ('forest', 'is', 'animate'), ('forest', 'east', 'forest_path'), ('forest', 'in', 'forest'), ('forest', 'west', 'clearing'), ('mountains', 'is', 'animate'), ('mountains', 'in', 'forest'), ('clearing', 'north', 'forest_path'), ('pile', 'in', 'clearing'), ('pile', 'is', 'animate'), ('leaves', 'is', 'animate'), ('leaves', 'in', 'clearing'), ('east', 'is', 'animate'), ('east', 'in', 'forest'), ('egg', 'is', 'animate'), ('nest', 'is', 'animate')]

==========STEP 20==========

Observation:

Up a Tree You are about 10 feet above the ground nestled among some large branches. The nearest branch above you is above your reach. Beside you on the branch is a small birds nest. You hear in the distance the chirping of a song bird.

You are carrying A jewelencrusted egg You hear in the distance the chirping of a song bird .

Taken.

take egg

Action: west

[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', 'in', 'west'), ('west', 'west', 'north_of_house'), ('mailbox', 'is', 'animate'), ('mailbox', 'in', 'west'), ('you', 'is', 'you'), ('you', 'have', 'egg'), ('you', 'have', 'jewel'), ('you', 'have', 'encrusted'), ('north_of_house', 'north', 'west'), ('north_of_house', 'south', 'forest_path'), ('windows', 'in', 'north_of_house'), ('windows', 'is', 'animate'), ('trees', 'in', 'north_of_house'), ('trees', 'is', 'animate'), ('trees', 'in', 'forest'), ('path', 'is', 'animate'), ('path', 'in', 'north_of_house'), ('path', 'in', 'clearing'), ('north', 'is', 'animate'), ('north', 'in', 'north_of_house'), ('large', 'in', 'forest_path'), ('large', 'is', 'animate'), ('large', 'in', 'forest'), ('forest_path', 'north', 'north_of_house'), ('forest_path', 'west', 'forest'), ('forest_path', 'south', 'clearing'), ('tree', 'in', 'forest_path'), ('tree', 'is', 'animate'), ('south', 'is', 'animate'), ('south', 'in', 'forest_path'), ('forest', 'is', 'animate'), ('forest', 'east', 'forest_path'), ('forest', 'in', 'forest'), ('forest', 'west', 'clearing'), ('mountains', 'is', 'animate'), ('mountains', 'in', 'forest'), ('clearing', 'north', 'forest_path'), ('pile', 'in', 'clearing'), ('pile', 'is', 'animate'), ('leaves', 'is', 'animate'), ('leaves', 'in', 'clearing'), ('east', 'is', 'animate'), ('east', 'in', 'forest'), ('egg', 'is', 'animate'), ('nest', 'is', 'animate'), ('jewel', 'is', 'animate'), ('encrusted', 'is', 'animate')]

==========STEP 21==========

Observation:

Up a Tree You are about 10 feet above the ground nestled among some large branches. The nearest branch above you is above your reach. Beside you on the branch is a small birds nest.

```
You are carrying A jewelencrusted egg
You cant go that way. You hear in the distance the chirping of a song bird.
west

Action: throw trees at window
[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', '
    in', 'west'), ('west', 'west', 'north_of_house'), ('mailbox', 'is', 'animate'), ('
    mailbox', 'in', 'west'), ('you', 'is', 'you'), ('you', 'have', 'egg'), ('you', 'have',
     'jewel'), ('you', 'have', 'encrusted'), ('north_of_house', 'north', 'west'), ('
    north_of_house', 'south', 'forest_path'), ('windows', 'in', 'north_of_house'), ('
    windows', 'is', 'animate'), ('trees', 'in', 'north_of_house'), ('trees', 'is', '
    animate'), ('trees', 'in', 'forest'), ('path', 'is', 'animate'), ('path', 'in', '
    north_of_house'), ('path', 'in', 'clearing'), ('north', 'is', 'animate'), ('north', '
    in', 'north_of_house'), ('large', 'in', 'forest_path'), ('large', 'is', 'animate'), ('
    large', 'in', 'forest'), ('forest_path', 'north', 'north_of_house'), ('forest_path', '
    west', 'forest'), ('forest_path', 'south', 'clearing'), ('tree', 'in', 'forest_path'),
     ('tree', 'is', 'animate'), ('south', 'is', 'animate'), ('south', 'in', 'forest_path')
    , ('forest', 'is', 'animate'), ('forest', 'east', 'forest_path'), ('forest', 'in', '
    forest'), ('forest', 'west', 'clearing'), ('mountains', 'is', 'animate'), ('mountains
    ', 'in', 'forest'), ('clearing', 'north', 'forest_path'), ('pile', 'in', 'clearing'),
    ('pile', 'is', 'animate'), ('leaves', 'is', 'animate'), ('leaves', 'in', 'clearing'),
    ('east', 'is', 'animate'), ('east', 'in', 'forest'), ('egg', 'is', 'animate'), ('nest
    ', 'is', 'animate'), ('jewel', 'is', 'animate'), ('encrusted', 'is', 'animate')]
==========STEP 22==========
Observation:
Up a Tree You are about 10 feet above the ground nestled among some large branches. The
    nearest branch above you is above your reach. Beside you on the branch is a small
    birds nest.
You are carrying A jewelencrusted egg
You dont have the forest.
throw trees at window

Action: throw forest at forest
[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', '
    in', 'west'), ('west', 'west', 'north_of_house'), ('mailbox', 'is', 'animate'), ('
    mailbox', 'in', 'west'), ('you', 'is', 'you'), ('you', 'have', 'egg'), ('you', 'have',
     'jewel'), ('you', 'have', 'encrusted'), ('north_of_house', 'north', 'west'), ('
    north_of_house', 'south', 'forest_path'), ('windows', 'in', 'north_of_house'), ('
    windows', 'is', 'animate'), ('trees', 'in', 'north_of_house'), ('trees', 'is', '
    animate'), ('trees', 'in', 'forest'), ('path', 'is', 'animate'), ('path', 'in', '
    north_of_house'), ('path', 'in', 'clearing'), ('north', 'is', 'animate'), ('north', '
```

```
    in', 'north_of_house'), ('large', 'in', 'forest_path'), ('large', 'is', 'animate'), ('
    large', 'in', 'forest'), ('forest_path', 'north', 'north_of_house'), ('forest_path', '
    west', 'forest'), ('forest_path', 'south', 'clearing'), ('tree', 'in', 'forest_path'),
     ('tree', 'is', 'animate'), ('south', 'is', 'animate'), ('south', 'in', 'forest_path')
    , ('forest', 'is', 'animate'), ('forest', 'east', 'forest_path'), ('forest', 'in', '
    forest'), ('forest', 'west', 'clearing'), ('mountains', 'is', 'animate'), ('mountains
    ', 'in', 'forest'), ('clearing', 'north', 'forest_path'), ('pile', 'in', 'clearing'),
    ('pile', 'is', 'animate'), ('leaves', 'is', 'animate'), ('leaves', 'in', 'clearing'),
    ('east', 'is', 'animate'), ('east', 'in', 'forest'), ('egg', 'is', 'animate'), ('nest
    ', 'is', 'animate'), ('jewel', 'is', 'animate'), ('encrusted', 'is', 'animate')]
==========STEP 23==========
Observation:
Up a Tree You are about 10 feet above the ground nestled among some large branches. The
    nearest branch above you is above your reach. Beside you on the branch is a small
    birds nest.
You are carrying A jewelencrusted egg
You dont have the forest.
throw forest at forest

Action: throw egg at nest
[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', '
    in', 'west'), ('west', 'west', 'north_of_house'), ('mailbox', 'is', 'animate'), ('
    mailbox', 'in', 'west'), ('you', 'is', 'you'), ('you', 'have', 'egg'), ('you', 'have',
     'jewel'), ('you', 'have', 'encrusted'), ('north_of_house', 'north', 'west'), ('
    north_of_house', 'south', 'forest_path'), ('windows', 'in', 'north_of_house'), ('
    windows', 'is', 'animate'), ('trees', 'in', 'north_of_house'), ('trees', 'is', '
    animate'), ('trees', 'in', 'forest'), ('path', 'is', 'animate'), ('path', 'in', '
    north_of_house'), ('path', 'in', 'clearing'), ('north', 'is', 'animate'), ('north', '
    in', 'north_of_house'), ('large', 'in', 'forest_path'), ('large', 'is', 'animate'), ('
    large', 'in', 'forest'), ('forest_path', 'north', 'north_of_house'), ('forest_path', '
    west', 'forest'), ('forest_path', 'south', 'clearing'), ('tree', 'in', 'forest_path'),
     ('tree', 'is', 'animate'), ('south', 'is', 'animate'), ('south', 'in', 'forest_path')
    , ('forest', 'is', 'animate'), ('forest', 'east', 'forest_path'), ('forest', 'in', '
    forest'), ('forest', 'west', 'clearing'), ('mountains', 'is', 'animate'), ('mountains
    ', 'in', 'forest'), ('clearing', 'north', 'forest_path'), ('pile', 'in', 'clearing'),
    ('pile', 'is', 'animate'), ('leaves', 'is', 'animate'), ('leaves', 'in', 'clearing'),
    ('east', 'is', 'animate'), ('east', 'in', 'forest'), ('egg', 'is', 'animate'), ('nest
    ', 'is', 'animate'), ('jewel', 'is', 'animate'), ('encrusted', 'is', 'animate')]
==========STEP 24==========
Observation:
Up a Tree You are about 10 feet above the ground nestled among some large branches. The
```

nearest branch above you is above your reach. Beside you on the branch is a small birds nest. There is a somewhat ruined egg here. There is a golden clockwork canary nestled in the egg. It seems to have recently had a bad experience. The mountings for its jewellike eyes are empty, and its silver beak is crumpled. Through a cracked crystal window below its left wing you can see the remains of intricate machinery. It is not clear what result winding it would have, as the mainspring seems sprung.

You are emptyhanded.

Your rather indelicate handling of the egg has caused it some damage, although you have succeeded in opening it. There is a golden clockwork canary nestled in the egg. It seems to have recently had a bad experience. The mountings for its jewellike eyes are empty, and its silver beak is crumpled. Through a cracked crystal window below its left wing you can see the remains of intricate machinery. It is not clear what result winding it would have, as the mainspring seems sprung.

throw egg at nest

Action: south

[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', 'in', 'west'), ('west', 'west', 'north_of_house'), ('mailbox', 'is', 'animate'), ('mailbox', 'in', 'west'), ('you', 'is', 'you'), ('you', 'have', 'jewel'), ('you', 'have', 'encrusted'), ('north_of_house', 'north', 'west'), ('north_of_house', 'south', 'forest_path'), ('windows', 'in', 'north_of_house'), ('windows', 'is', 'animate'), ('trees', 'in', 'north_of_house'), ('trees', 'is', 'animate'), ('trees', 'in', 'forest'), ('path', 'is', 'animate'), ('path', 'in', 'north_of_house'), ('path', 'in', 'clearing'), ('north', 'is', 'animate'), ('north', 'in', 'north_of_house'), ('large', 'in', 'forest_path'), ('large', 'is', 'animate'), ('large', 'in', 'forest'), ('forest_path', 'north', 'north_of_house'), ('forest_path', 'west', 'forest'), ('forest_path', 'south', 'clearing'), ('tree', 'in', 'forest_path'), ('tree', 'is', 'animate'), ('south', 'is', 'animate'), ('south', 'in', 'forest_path'), ('forest', 'is', 'animate'), ('forest', 'east', 'forest_path'), ('forest', 'in', 'forest'), ('forest', 'west', 'clearing'), ('mountains', 'is', 'animate'), ('mountains', 'in', 'forest'), ('clearing', 'north', 'forest_path'), ('pile', 'in', 'clearing'), ('pile', 'is', 'animate'), ('leaves', 'is', 'animate'), ('leaves', 'in', 'clearing'), ('east', 'is', 'animate'), ('east', 'in', 'forest'), ('egg', 'is', 'animate'), ('nest', 'is', 'animate'), ('jewel', 'is', 'animate'), ('encrusted', 'is', 'animate'), ('canary', 'is', 'animate'), ('window', 'is', 'animate')]

==========STEP 25==========

Observation:

Up a Tree You are about 10 feet above the ground nestled among some large branches. The nearest branch above you is above your reach. Beside you on the branch is a small birds nest. There is a somewhat ruined egg here. There is a golden clockwork canary nestled in the egg. It seems to have recently had a bad experience. The mountings for

its jewellike eyes are empty, and its silver beak is crumpled. Through a cracked crystal window below its left wing you can see the remains of intricate machinery. It is not clear what result winding it would have, as the mainspring seems sprung.

You are emptyhanded.

You cant go that way.

south

Action: take egg

[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', 'in', 'west'), ('west', 'west', 'north_of_house'), ('mailbox', 'is', 'animate'), ('mailbox', 'in', 'west'), ('you', 'is', 'you'), ('you', 'have', 'jewel'), ('you', 'have', 'encrusted'), ('north_of_house', 'north', 'west'), ('north_of_house', 'south', 'forest_path'), ('windows', 'in', 'north_of_house'), ('windows', 'is', 'animate'), ('trees', 'in', 'north_of_house'), ('trees', 'is', 'animate'), ('trees', 'in', 'forest'), ('path', 'is', 'animate'), ('path', 'in', 'north_of_house'), ('path', 'in', 'clearing'), ('north', 'is', 'animate'), ('north', 'in', 'north_of_house'), ('large', 'in', 'forest_path'), ('large', 'is', 'animate'), ('large', 'in', 'forest'), ('forest_path', 'north', 'north_of_house'), ('forest_path', 'west', 'forest'), ('forest_path', 'south', 'clearing'), ('tree', 'in', 'forest_path'), ('tree', 'is', 'animate'), ('south', 'is', 'animate'), ('south', 'in', 'forest_path'), ('forest', 'is', 'animate'), ('forest', 'east', 'forest_path'), ('forest', 'in', 'forest'), ('forest', 'west', 'clearing'), ('mountains', 'is', 'animate'), ('mountains', 'in', 'forest'), ('clearing', 'north', 'forest_path'), ('pile', 'in', 'clearing'), ('pile', 'is', 'animate'), ('leaves', 'is', 'animate'), ('leaves', 'in', 'clearing'), ('east', 'is', 'animate'), ('east', 'in', 'forest'), ('egg', 'is', 'animate'), ('nest', 'is', 'animate'), ('jewel', 'is', 'animate'), ('encrusted', 'is', 'animate'), ('canary', 'is', 'animate'), ('window', 'is', 'animate')]

==========STEP 26==========

Observation:

Up a Tree You are about 10 feet above the ground nestled among some large branches. The nearest branch above you is above your reach. Beside you on the branch is a small birds nest.

You are carrying A broken jewelencrusted egg The broken jewelencrusted egg contains A broken clockwork canary

Taken.

take egg

Action: throw egg at nest

[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', 'in', 'west'), ('west', 'west', 'north_of_house'), ('mailbox', 'is', 'animate'), ('mailbox', 'in', 'west'), ('you', 'is', 'you'), ('you', 'have', 'jewel'), ('you', 'have

', 'encrusted'), ('you', 'have', 'clockwork'), ('you', 'have', 'broken'), ('north_of_house', 'north', 'west'), ('north_of_house', 'south', 'forest_path'), ('windows', 'in', 'north_of_house'), ('windows', 'is', 'animate'), ('trees', 'in', 'north_of_house'), ('trees', 'is', 'animate'), ('trees', 'in', 'forest'), ('path', 'is', 'animate'), ('path', 'in', 'north_of_house'), ('path', 'in', 'clearing'), ('north', 'is', 'animate'), ('north', 'in', 'north_of_house'), ('large', 'in', 'forest_path'), ('large', 'is', 'animate'), ('large', 'in', 'forest'), ('forest_path', 'north', 'north_of_house'), ('forest_path', 'west', 'forest'), ('forest_path', 'south', 'clearing'), ('tree', 'in', 'forest_path'), ('tree', 'is', 'animate'), ('south', 'is', 'animate'), ('south', 'in', 'forest_path'), ('forest', 'is', 'animate'), ('forest', 'east', 'forest_path'), ('forest', 'in', 'forest'), ('forest', 'west', 'clearing'), ('mountains', 'is', 'animate'), ('mountains', 'in', 'forest'), ('clearing', 'north', 'forest_path'), ('pile', 'in', 'clearing'), ('pile', 'is', 'animate'), ('leaves', 'is', 'animate'), ('leaves', 'in', 'clearing'), ('east', 'is', 'animate'), ('east', 'in', 'forest'), ('egg', 'is', 'animate'), ('nest', 'is', 'animate'), ('jewel', 'is', 'animate'), ('encrusted', 'is', 'animate'), ('canary', 'is', 'animate'), ('window', 'is', 'animate'), ('broken', 'is', 'animate'), ('clockwork', 'is', 'animate')]

==========STEP 27==========

Observation:

Up a Tree You are about 10 feet above the ground nestled among some large branches. The nearest branch above you is above your reach. Beside you on the branch is a small birds nest. There is a somewhat ruined egg here. There is a golden clockwork canary nestled in the egg. It seems to have recently had a bad experience. The mountings for its jewellike eyes are empty, and its silver beak is crumpled. Through a cracked crystal window below its left wing you can see the remains of intricate machinery. It is not clear what result winding it would have, as the mainspring seems sprung.

You are emptyhanded.

Thrown.

throw egg at nest

Action: open egg

[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', 'in', 'west'), ('west', 'west', 'north_of_house'), ('mailbox', 'is', 'animate'), ('mailbox', 'in', 'west'), ('you', 'is', 'you'), ('you', 'have', 'jewel'), ('you', 'have', 'encrusted'), ('you', 'have', 'clockwork'), ('you', 'have', 'broken'), ('north_of_house', 'north', 'west'), ('north_of_house', 'south', 'forest_path'), ('windows', 'in', 'north_of_house'), ('windows', 'is', 'animate'), ('trees', 'in', 'north_of_house'), ('trees', 'is', 'animate'), ('trees', 'in', 'forest'), ('path', 'is', 'animate'), ('path', 'in', 'north_of_house'), ('path', 'in', 'clearing'), ('north', 'is', 'animate'), ('north', 'in', 'north_of_house'), ('large', 'in', 'forest_path'), ('large', 'is', 'animate'), ('large', 'in', 'forest'), ('forest_path', 'north', '

north_of_house'), ('forest_path', 'west', 'forest'), ('forest_path', 'south', '
clearing'), ('tree', 'in', 'forest_path'), ('tree', 'is', 'animate'), ('south', 'is',
'animate'), ('south', 'in', 'forest_path'), ('forest', 'is', 'animate'), ('forest', '
east', 'forest_path'), ('forest', 'in', 'forest'), ('forest', 'west', 'clearing'), ('
mountains', 'is', 'animate'), ('mountains', 'in', 'forest'), ('clearing', 'north', '
forest_path'), ('pile', 'in', 'clearing'), ('pile', 'is', 'animate'), ('leaves', 'is',
'animate'), ('leaves', 'in', 'clearing'), ('east', 'is', 'animate'), ('east', 'in', '
forest'), ('egg', 'is', 'animate'), ('nest', 'is', 'animate'), ('jewel', 'is', '
animate'), ('encrusted', 'is', 'animate'), ('canary', 'is', 'animate'), ('window', 'is
', 'animate'), ('broken', 'is', 'animate'), ('clockwork', 'is', 'animate')]

==========STEP 28==========

Observation:

Up a Tree You are about 10 feet above the ground nestled among some large branches. The nearest branch above you is above your reach. Beside you on the branch is a small birds nest. There is a somewhat ruined egg here. There is a golden clockwork canary nestled in the egg. It seems to have recently had a bad experience. The mountings for its jewellike eyes are empty, and its silver beak is crumpled. Through a cracked crystal window below its left wing you can see the remains of intricate machinery. It is not clear what result winding it would have, as the mainspring seems sprung. You hear in the distance the chirping of a song bird.

You are emptyhanded. You hear in the distance the chirping of a song bird.

It is already open.

open egg

Action: north

[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', '
in', 'west'), ('west', 'west', 'north_of_house'), ('mailbox', 'is', 'animate'), ('
mailbox', 'in', 'west'), ('you', 'is', 'you'), ('you', 'have', 'jewel'), ('you', 'have
', 'encrusted'), ('you', 'have', 'clockwork'), ('you', 'have', 'broken'), ('
north_of_house', 'north', 'west'), ('north_of_house', 'south', 'forest_path'), ('
windows', 'in', 'north_of_house'), ('windows', 'is', 'animate'), ('trees', 'in', '
north_of_house'), ('trees', 'is', 'animate'), ('trees', 'in', 'forest'), ('path', 'is
', 'animate'), ('path', 'in', 'north_of_house'), ('path', 'in', 'clearing'), ('north',
'is', 'animate'), ('north', 'in', 'north_of_house'), ('large', 'in', 'forest_path'),
('large', 'is', 'animate'), ('large', 'in', 'forest'), ('forest_path', 'north', '
north_of_house'), ('forest_path', 'west', 'forest'), ('forest_path', 'south', '
clearing'), ('tree', 'in', 'forest_path'), ('tree', 'is', 'animate'), ('south', 'is',
'animate'), ('south', 'in', 'forest_path'), ('forest', 'is', 'animate'), ('forest', '
east', 'forest_path'), ('forest', 'in', 'forest'), ('forest', 'west', 'clearing'), ('
mountains', 'is', 'animate'), ('mountains', 'in', 'forest'), ('clearing', 'north', '
forest_path'), ('pile', 'in', 'clearing'), ('pile', 'is', 'animate'), ('leaves', 'is',

'animate'), ('leaves', 'in', 'clearing'), ('east', 'is', 'animate'), ('east', 'in', '
forest'), ('egg', 'is', 'animate'), ('nest', 'is', 'animate'), ('jewel', 'is', '
animate'), ('encrusted', 'is', 'animate'), ('canary', 'is', 'animate'), ('window', 'is
', 'animate'), ('broken', 'is', 'animate'), ('clockwork', 'is', 'animate')]

==========STEP 29==========

Observation:

Up a Tree You are about 10 feet above the ground nestled among some large branches. The nearest branch above you is above your reach. Beside you on the branch is a small birds nest. There is a somewhat ruined egg here. There is a golden clockwork canary nestled in the egg. It seems to have recently had a bad experience. The mountings for its jewellike eyes are empty, and its silver beak is crumpled. Through a cracked crystal window below its left wing you can see the remains of intricate machinery. It is not clear what result winding it would have, as the mainspring seems sprung.

You are emptyhanded.

You cant go that way. You hear in the distance the chirping of a song bird.

north

Action: close nest

[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', '
in', 'west'), ('west', 'west', 'north_of_house'), ('mailbox', 'is', 'animate'), ('
mailbox', 'in', 'west'), ('you', 'is', 'you'), ('you', 'have', 'jewel'), ('you', 'have
', 'encrusted'), ('you', 'have', 'clockwork'), ('you', 'have', 'broken'), ('
north_of_house', 'north', 'west'), ('north_of_house', 'south', 'forest_path'), ('
windows', 'in', 'north_of_house'), ('windows', 'is', 'animate'), ('trees', 'in', '
north_of_house'), ('trees', 'is', 'animate'), ('trees', 'in', 'forest'), ('path', 'is
', 'animate'), ('path', 'in', 'north_of_house'), ('path', 'in', 'clearing'), ('north',
'is', 'animate'), ('north', 'in', 'north_of_house'), ('large', 'in', 'forest_path'),
('large', 'is', 'animate'), ('large', 'in', 'forest'), ('forest_path', 'north', '
north_of_house'), ('forest_path', 'west', 'forest'), ('forest_path', 'south', '
clearing'), ('tree', 'in', 'forest_path'), ('tree', 'is', 'animate'), ('south', 'is',
'animate'), ('south', 'in', 'forest_path'), ('forest', 'is', 'animate'), ('forest', '
east', 'forest_path'), ('forest', 'in', 'forest'), ('forest', 'west', 'clearing'), ('
mountains', 'is', 'animate'), ('mountains', 'in', 'forest'), ('clearing', 'north', '
forest_path'), ('pile', 'in', 'clearing'), ('pile', 'is', 'animate'), ('leaves', 'is',
'animate'), ('leaves', 'in', 'clearing'), ('east', 'is', 'animate'), ('east', 'in', '
forest'), ('egg', 'is', 'animate'), ('nest', 'is', 'animate'), ('jewel', 'is', '
animate'), ('encrusted', 'is', 'animate'), ('canary', 'is', 'animate'), ('window', 'is
', 'animate'), ('broken', 'is', 'animate'), ('clockwork', 'is', 'animate'), ('
clockwork_canary', 'is', 'animate')]

==========STEP 30==========

Observation:

Up a Tree You are about 10 feet above the ground nestled among some large branches. The nearest branch above you is above your reach. Beside you on the branch is a small birds nest. There is a somewhat ruined egg here. There is a golden clockwork canary nestled in the egg. It seems to have recently had a bad experience. The mountings for its jewellike eyes are empty, and its silver beak is crumpled. Through a cracked crystal window below its left wing you can see the remains of intricate machinery. It is not clear what result winding it would have, as the mainspring seems sprung.
You are emptyhanded.
Closed.
close nest

Action: east
[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', 'in', 'west'), ('west', 'west', 'north_of_house'), ('mailbox', 'is', 'animate'), ('mailbox', 'in', 'west'), ('you', 'is', 'you'), ('you', 'have', 'jewel'), ('you', 'have', 'encrusted'), ('you', 'have', 'clockwork'), ('you', 'have', 'broken'), ('north_of_house', 'north', 'west'), ('north_of_house', 'south', 'forest_path'), ('windows', 'in', 'north_of_house'), ('windows', 'is', 'animate'), ('trees', 'in', 'north_of_house'), ('trees', 'is', 'animate'), ('trees', 'in', 'forest'), ('path', 'is', 'animate'), ('path', 'in', 'north_of_house'), ('path', 'in', 'clearing'), ('north', 'is', 'animate'), ('north', 'in', 'north_of_house'), ('large', 'in', 'forest_path'), ('large', 'is', 'animate'), ('large', 'in', 'forest'), ('forest_path', 'north', 'north_of_house'), ('forest_path', 'west', 'forest'), ('forest_path', 'south', 'clearing'), ('tree', 'in', 'forest_path'), ('tree', 'is', 'animate'), ('south', 'is', 'animate'), ('south', 'in', 'forest_path'), ('forest', 'is', 'animate'), ('forest', 'east', 'forest_path'), ('forest', 'in', 'forest'), ('forest', 'west', 'clearing'), ('mountains', 'is', 'animate'), ('mountains', 'in', 'forest'), ('clearing', 'north', 'forest_path'), ('pile', 'in', 'clearing'), ('pile', 'is', 'animate'), ('leaves', 'is', 'animate'), ('leaves', 'in', 'clearing'), ('east', 'is', 'animate'), ('east', 'in', 'forest'), ('egg', 'is', 'animate'), ('nest', 'is', 'animate'), ('jewel', 'is', 'animate'), ('encrusted', 'is', 'animate'), ('canary', 'is', 'animate'), ('window', 'is', 'animate'), ('broken', 'is', 'animate'), ('clockwork', 'is', 'animate'), ('clockwork_canary', 'is', 'animate')]
==========STEP 31==========
Observation:
Up a Tree You are about 10 feet above the ground nestled among some large branches. The nearest branch above you is above your reach. Beside you on the branch is a small birds nest. There is a somewhat ruined egg here. There is a golden clockwork canary nestled in the egg. It seems to have recently had a bad experience. The mountings for its jewellike eyes are empty, and its silver beak is crumpled. Through a cracked crystal window below its left wing you can see the remains of intricate machinery. It

is not clear what result winding it would have, as the mainspring seems sprung.
You are emptyhanded.
You cant go that way.
east

Action: north
[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', '
in', 'west'), ('west', 'west', 'north_of_house'), ('mailbox', 'is', 'animate'), ('
mailbox', 'in', 'west'), ('you', 'is', 'you'), ('you', 'have', 'jewel'), ('you', 'have
', 'encrusted'), ('you', 'have', 'clockwork'), ('you', 'have', 'broken'), ('
north_of_house', 'north', 'west'), ('north_of_house', 'south', 'forest_path'), ('
windows', 'in', 'north_of_house'), ('windows', 'is', 'animate'), ('trees', 'in', '
north_of_house'), ('trees', 'is', 'animate'), ('trees', 'in', 'forest'), ('path', 'is
', 'animate'), ('path', 'in', 'north_of_house'), ('path', 'in', 'clearing'), ('north',
 'is', 'animate'), ('north', 'in', 'north_of_house'), ('large', 'in', 'forest_path'),
('large', 'is', 'animate'), ('large', 'in', 'forest'), ('forest_path', 'north', '
north_of_house'), ('forest_path', 'west', 'forest'), ('forest_path', 'south', '
clearing'), ('tree', 'in', 'forest_path'), ('tree', 'is', 'animate'), ('south', 'is',
'animate'), ('south', 'in', 'forest_path'), ('forest', 'is', 'animate'), ('forest', '
east', 'forest_path'), ('forest', 'in', 'forest'), ('forest', 'west', 'clearing'), ('
mountains', 'is', 'animate'), ('mountains', 'in', 'forest'), ('clearing', 'north', '
forest_path'), ('pile', 'in', 'clearing'), ('pile', 'is', 'animate'), ('leaves', 'is',
 'animate'), ('leaves', 'in', 'clearing'), ('east', 'is', 'animate'), ('east', 'in', '
forest'), ('egg', 'is', 'animate'), ('nest', 'is', 'animate'), ('jewel', 'is', '
animate'), ('encrusted', 'is', 'animate'), ('canary', 'is', 'animate'), ('window', 'is
', 'animate'), ('broken', 'is', 'animate'), ('clockwork', 'is', 'animate'), ('
clockwork_canary', 'is', 'animate')]
==========STEP 32==========
Observation:
Up a Tree You are about 10 feet above the ground nestled among some large branches. The
nearest branch above you is above your reach. Beside you on the branch is a small
birds nest. There is a somewhat ruined egg here. There is a golden clockwork canary
nestled in the egg. It seems to have recently had a bad experience. The mountings for
its jewellike eyes are empty, and its silver beak is crumpled. Through a cracked
crystal window below its left wing you can see the remains of intricate machinery. It
is not clear what result winding it would have, as the mainspring seems sprung.
You are emptyhanded.
You cant go that way.
north

Action: take egg

[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', '
    in', 'west'), ('west', 'west', 'north_of_house'), ('mailbox', 'is', 'animate'), ('
    mailbox', 'in', 'west'), ('you', 'is', 'you'), ('you', 'have', 'jewel'), ('you', 'have
    ', 'encrusted'), ('you', 'have', 'clockwork'), ('you', 'have', 'broken'), ('
    north_of_house', 'north', 'west'), ('north_of_house', 'south', 'forest_path'), ('
    windows', 'in', 'north_of_house'), ('windows', 'is', 'animate'), ('trees', 'in', '
    north_of_house'), ('trees', 'is', 'animate'), ('trees', 'in', 'forest'), ('path', 'is
    ', 'animate'), ('path', 'in', 'north_of_house'), ('path', 'in', 'clearing'), ('north',
     'is', 'animate'), ('north', 'in', 'north_of_house'), ('large', 'in', 'forest_path'),
    ('large', 'is', 'animate'), ('large', 'in', 'forest'), ('forest_path', 'north', '
    north_of_house'), ('forest_path', 'west', 'forest'), ('forest_path', 'south', '
    clearing'), ('tree', 'in', 'forest_path'), ('tree', 'is', 'animate'), ('south', 'is',
    'animate'), ('south', 'in', 'forest_path'), ('forest', 'is', 'animate'), ('forest', '
    east', 'forest_path'), ('forest', 'in', 'forest'), ('forest', 'west', 'clearing'), ('
    mountains', 'is', 'animate'), ('mountains', 'in', 'forest'), ('clearing', 'north', '
    forest_path'), ('pile', 'in', 'clearing'), ('pile', 'is', 'animate'), ('leaves', 'is',
     'animate'), ('leaves', 'in', 'clearing'), ('east', 'is', 'animate'), ('east', 'in', '
    forest'), ('egg', 'is', 'animate'), ('nest', 'is', 'animate'), ('jewel', 'is', '
    animate'), ('encrusted', 'is', 'animate'), ('canary', 'is', 'animate'), ('window', 'is
    ', 'animate'), ('broken', 'is', 'animate'), ('clockwork', 'is', 'animate'), ('
    clockwork_canary', 'is', 'animate')]
==========STEP 33==========
Observation:
Up a Tree You are about 10 feet above the ground nestled among some large branches. The
    nearest branch above you is above your reach. Beside you on the branch is a small
    birds nest.
You are carrying A broken jewelencrusted egg The broken jewelencrusted egg contains A
    broken clockwork canary
Taken.
take egg

Action: shake egg
[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', '
    in', 'west'), ('west', 'west', 'north_of_house'), ('mailbox', 'is', 'animate'), ('
    mailbox', 'in', 'west'), ('you', 'is', 'you'), ('you', 'have', 'jewel'), ('you', 'have
    ', 'encrusted'), ('you', 'have', 'clockwork'), ('you', 'have', 'broken'), ('
    north_of_house', 'north', 'west'), ('north_of_house', 'south', 'forest_path'), ('
    windows', 'in', 'north_of_house'), ('windows', 'is', 'animate'), ('trees', 'in', '
    north_of_house'), ('trees', 'is', 'animate'), ('trees', 'in', 'forest'), ('path', 'is
    ', 'animate'), ('path', 'in', 'north_of_house'), ('path', 'in', 'clearing'), ('north',
     'is', 'animate'), ('north', 'in', 'north_of_house'), ('large', 'in', 'forest_path'),

('large', 'is', 'animate'), ('large', 'in', 'forest'), ('forest_path', 'north', '
north_of_house'), ('forest_path', 'west', 'forest'), ('forest_path', 'south', '
clearing'), ('tree', 'in', 'forest_path'), ('tree', 'is', 'animate'), ('south', 'is',
'animate'), ('south', 'in', 'forest_path'), ('forest', 'is', 'animate'), ('forest', '
east', 'forest_path'), ('forest', 'in', 'forest'), ('forest', 'west', 'clearing'), ('
mountains', 'is', 'animate'), ('mountains', 'in', 'forest'), ('clearing', 'north', '
forest_path'), ('pile', 'in', 'clearing'), ('pile', 'is', 'animate'), ('leaves', 'is',
'animate'), ('leaves', 'in', 'clearing'), ('east', 'is', 'animate'), ('east', 'in', '
forest'), ('egg', 'is', 'animate'), ('nest', 'is', 'animate'), ('jewel', 'is', '
animate'), ('encrusted', 'is', 'animate'), ('canary', 'is', 'animate'), ('window', 'is
', 'animate'), ('broken', 'is', 'animate'), ('clockwork', 'is', 'animate'), ('
clockwork_canary', 'is', 'animate')]

==========STEP 34==========

Observation:

Up a Tree You are about 10 feet above the ground nestled among some large branches. The
nearest branch above you is above your reach. Beside you on the branch is a small
birds nest.

You are carrying A broken jewelencrusted egg

The contents of the broken jewelencrusted egg spills to the ground.

shake egg

Action: down

[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', '
in', 'west'), ('west', 'west', 'north_of_house'), ('mailbox', 'is', 'animate'), ('
mailbox', 'in', 'west'), ('you', 'is', 'you'), ('you', 'have', 'jewel'), ('you', 'have
', 'encrusted'), ('you', 'have', 'clockwork'), ('you', 'have', 'broken'), ('you', '
have', 'egg'), ('north_of_house', 'north', 'west'), ('north_of_house', 'south', '
forest_path'), ('windows', 'in', 'north_of_house'), ('windows', 'is', 'animate'), ('
trees', 'in', 'north_of_house'), ('trees', 'is', 'animate'), ('trees', 'in', 'forest')
, ('path', 'is', 'animate'), ('path', 'in', 'north_of_house'), ('path', 'in', '
clearing'), ('north', 'is', 'animate'), ('north', 'in', 'north_of_house'), ('large', '
in', 'forest_path'), ('large', 'is', 'animate'), ('large', 'in', 'forest'), ('
forest_path', 'north', 'north_of_house'), ('forest_path', 'west', 'forest'), ('
forest_path', 'south', 'clearing'), ('tree', 'in', 'forest_path'), ('tree', 'is', '
animate'), ('south', 'is', 'animate'), ('south', 'in', 'forest_path'), ('forest', 'is
', 'animate'), ('forest', 'east', 'forest_path'), ('forest', 'in', 'forest'), ('forest
', 'west', 'clearing'), ('mountains', 'is', 'animate'), ('mountains', 'in', 'forest'),
('clearing', 'north', 'forest_path'), ('pile', 'in', 'clearing'), ('pile', 'is', '
animate'), ('leaves', 'is', 'animate'), ('leaves', 'in', 'clearing'), ('east', 'is', '
animate'), ('east', 'in', 'forest'), ('egg', 'is', 'animate'), ('nest', 'is', 'animate
'), ('jewel', 'is', 'animate'), ('encrusted', 'is', 'animate'), ('canary', 'is', '

animate'), ('window', 'is', 'animate'), ('broken', 'is', 'animate'), ('clockwork', 'is', 'animate'), ('clockwork_canary', 'is', 'animate'), ('ground', 'is', 'animate')]

==========STEP 35==========

Observation:

Forest Path This is a path winding through a dimly lit forest. The path heads northsouth here. One particularly large tree with some low branches stands at the edge of the path. There is a broken clockwork canary here.

You are carrying A broken jewelencrusted egg

Forest Path There is a broken clockwork canary here.

down

Action: east

[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', 'in', 'west'), ('west', 'west', 'north_of_house'), ('mailbox', 'is', 'animate'), ('mailbox', 'in', 'west'), ('you', 'is', 'you'), ('you', 'have', 'jewel'), ('you', 'have', 'encrusted'), ('you', 'have', 'egg'), ('you', 'in', 'forest_path'), ('north_of_house', 'north', 'west'), ('north_of_house', 'south', 'forest_path'), ('windows', 'in', 'north_of_house'), ('windows', 'is', 'animate'), ('trees', 'in', 'north_of_house'), ('trees', 'is', 'animate'), ('trees', 'in', 'forest'), ('path', 'is', 'animate'), ('path', 'in', 'north_of_house'), ('path', 'in', 'clearing'), ('north', 'is', 'animate'), ('north', 'in', 'north_of_house'), ('large', 'in', 'forest_path'), ('large', 'is', 'animate'), ('large', 'in', 'forest'), ('forest_path', 'north', 'north_of_house'), ('forest_path', 'west', 'forest'), ('forest_path', 'south', 'clearing'), ('tree', 'in', 'forest_path'), ('tree', 'is', 'animate'), ('south', 'is', 'animate'), ('south', 'in', 'forest_path'), ('forest', 'is', 'animate'), ('forest', 'east', 'forest_path'), ('forest', 'in', 'forest'), ('forest', 'west', 'clearing'), ('mountains', 'is', 'animate'), ('mountains', 'in', 'forest'), ('clearing', 'north', 'forest_path'), ('pile', 'in', 'clearing'), ('pile', 'is', 'animate'), ('leaves', 'is', 'animate'), ('leaves', 'in', 'clearing'), ('east', 'is', 'animate'), ('east', 'in', 'forest'), ('egg', 'is', 'animate'), ('nest', 'is', 'animate'), ('jewel', 'is', 'animate'), ('encrusted', 'is', 'animate'), ('canary', 'is', 'animate'), ('canary', 'in', 'forest_path'), ('window', 'is', 'animate'), ('broken', 'is', 'animate'), ('broken', 'in', 'forest_path'), ('clockwork', 'is', 'animate'), ('clockwork', 'in', 'forest_path'), ('clockwork_canary', 'is', 'animate'), ('clockwork_canary', 'in', 'forest_path'), ('ground', 'is', 'animate')]

==========STEP 36==========

Observation:

Forest This is a dimly lit forest, with large trees all around.

You are carrying A broken jewelencrusted egg

Forest

east

```
Action: go around egg
[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', '
    in', 'west'), ('west', 'west', 'north_of_house'), ('mailbox', 'is', 'animate'), ('
    mailbox', 'in', 'west'), ('you', 'is', 'you'), ('you', 'have', 'jewel'), ('you', 'have
    ', 'encrusted'), ('you', 'have', 'egg'), ('you', 'have', 'broken'), ('you', 'in', '
    forest'), ('north_of_house', 'north', 'west'), ('north_of_house', 'south', '
    forest_path'), ('windows', 'in', 'north_of_house'), ('windows', 'is', 'animate'), ('
    trees', 'in', 'north_of_house'), ('trees', 'is', 'animate'), ('trees', 'in', 'forest')
    , ('path', 'is', 'animate'), ('path', 'in', 'north_of_house'), ('path', 'in', '
    clearing'), ('north', 'is', 'animate'), ('north', 'in', 'north_of_house'), ('large', '
    in', 'forest_path'), ('large', 'is', 'animate'), ('large', 'in', 'forest'), ('
    forest_path', 'north', 'north_of_house'), ('forest_path', 'west', 'forest'), ('
    forest_path', 'south', 'clearing'), ('tree', 'in', 'forest_path'), ('tree', 'is', '
    animate'), ('south', 'is', 'animate'), ('south', 'in', 'forest_path'), ('forest', 'is
    ', 'animate'), ('forest', 'east', 'forest_path'), ('forest', 'in', 'forest'), ('forest
    ', 'west', 'clearing'), ('mountains', 'is', 'animate'), ('mountains', 'in', 'forest'),
     ('clearing', 'north', 'forest_path'), ('pile', 'in', 'clearing'), ('pile', 'is', '
    animate'), ('leaves', 'is', 'animate'), ('leaves', 'in', 'clearing'), ('east', 'is', '
    animate'), ('east', 'in', 'forest'), ('egg', 'is', 'animate'), ('nest', 'is', 'animate
    '), ('jewel', 'is', 'animate'), ('encrusted', 'is', 'animate'), ('canary', 'is', '
    animate'), ('canary', 'in', 'forest_path'), ('window', 'is', 'animate'), ('broken', '
    is', 'animate'), ('broken', 'in', 'forest_path'), ('clockwork', 'is', 'animate'), ('
    clockwork', 'in', 'forest_path'), ('clockwork_canary', 'is', 'animate'), ('
    clockwork_canary', 'in', 'forest_path'), ('ground', 'is', 'animate')]
==========STEP 37==========
Observation:
Forest This is a dimly lit forest, with large trees all around. You hear in the distance
    the chirping of a song bird.
You are carrying A broken jewelencrusted egg You hear in the distance the chirping of a
    song bird.
Use compass directions for movement.
go around egg

Action: put down nest
[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', '
    in', 'west'), ('west', 'west', 'north_of_house'), ('mailbox', 'is', 'animate'), ('
    mailbox', 'in', 'west'), ('you', 'is', 'you'), ('you', 'have', 'jewel'), ('you', 'have
    ', 'encrusted'), ('you', 'have', 'egg'), ('you', 'have', 'broken'), ('north_of_house',
     'north', 'west'), ('north_of_house', 'south', 'forest_path'), ('windows', 'in', '
    north_of_house'), ('windows', 'is', 'animate'), ('trees', 'in', 'north_of_house'), ('
```

trees', 'is', 'animate'), ('trees', 'in', 'forest'), ('path', 'is', 'animate'), ('path', 'in', 'north_of_house'), ('path', 'in', 'clearing'), ('north', 'is', 'animate'), ('north', 'in', 'north_of_house'), ('large', 'in', 'forest_path'), ('large', 'is', 'animate'), ('large', 'in', 'forest'), ('forest_path', 'north', 'north_of_house'), ('forest_path', 'west', 'forest'), ('forest_path', 'south', 'clearing'), ('tree', 'in', 'forest_path'), ('tree', 'is', 'animate'), ('south', 'is', 'animate'), ('south', 'in', 'forest_path'), ('forest', 'is', 'animate'), ('forest', 'east', 'forest_path'), ('forest', 'in', 'forest'), ('forest', 'west', 'clearing'), ('mountains', 'is', 'animate'), ('mountains', 'in', 'forest'), ('clearing', 'north', 'forest_path'), ('pile', 'in', 'clearing'), ('pile', 'is', 'animate'), ('leaves', 'is', 'animate'), ('leaves', 'in', 'clearing'), ('east', 'is', 'animate'), ('east', 'in', 'forest'), ('egg', 'is', 'animate'), ('nest', 'is', 'animate'), ('jewel', 'is', 'animate'), ('encrusted', 'is', 'animate'), ('canary', 'is', 'animate'), ('canary', 'in', 'forest_path'), ('window', 'is', 'animate'), ('broken', 'is', 'animate'), ('broken', 'in', 'forest_path'), ('clockwork', 'is', 'animate'), ('clockwork', 'in', 'forest_path'), ('clockwork_canary', 'is', 'animate'), ('clockwork_canary', 'in', 'forest_path'), ('ground', 'is', 'animate')]

==========STEP 38==========

Observation:

Forest This is a dimly lit forest, with large trees all around.

You are carrying A broken jewelencrusted egg

You cant see any nest here! You hear in the distance the chirping of a song bird.

put down nest

Action: go around egg

[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', 'in', 'west'), ('west', 'west', 'north_of_house'), ('mailbox', 'is', 'animate'), ('mailbox', 'in', 'west'), ('you', 'is', 'you'), ('you', 'have', 'jewel'), ('you', 'have', 'encrusted'), ('you', 'have', 'egg'), ('you', 'have', 'broken'), ('north_of_house', 'north', 'west'), ('north_of_house', 'south', 'forest_path'), ('windows', 'in', 'north_of_house'), ('windows', 'is', 'animate'), ('trees', 'in', 'north_of_house'), ('trees', 'is', 'animate'), ('trees', 'in', 'forest'), ('path', 'is', 'animate'), ('path', 'in', 'north_of_house'), ('path', 'in', 'clearing'), ('north', 'is', 'animate'), ('north', 'in', 'north_of_house'), ('large', 'in', 'forest_path'), ('large', 'is', 'animate'), ('large', 'in', 'forest'), ('forest_path', 'north', 'north_of_house'), ('forest_path', 'west', 'forest'), ('forest_path', 'south', 'clearing'), ('tree', 'in', 'forest_path'), ('tree', 'is', 'animate'), ('south', 'is', 'animate'), ('south', 'in', 'forest_path'), ('forest', 'is', 'animate'), ('forest', 'east', 'forest_path'), ('forest', 'in', 'forest'), ('forest', 'west', 'clearing'), ('mountains', 'is', 'animate'), ('mountains', 'in', 'forest'), ('clearing', 'north', 'forest_path'), ('pile', 'in', 'clearing'), ('pile', 'is', 'animate'), ('leaves', 'is', 'animate'), ('leaves', 'in

```
    ', 'clearing'), ('east', 'is', 'animate'), ('east', 'in', 'forest'), ('egg', 'is', '
    animate'), ('nest', 'is', 'animate'), ('jewel', 'is', 'animate'), ('encrusted', 'is',
    'animate'), ('canary', 'is', 'animate'), ('canary', 'in', 'forest_path'), ('window', '
    is', 'animate'), ('broken', 'is', 'animate'), ('broken', 'in', 'forest_path'), ('
    clockwork', 'is', 'animate'), ('clockwork', 'in', 'forest_path'), ('clockwork_canary',
     'is', 'animate'), ('clockwork_canary', 'in', 'forest_path'), ('ground', 'is', '
    animate')]
==========STEP 39==========
Observation:
Forest This is a dimly lit forest, with large trees all around.
You are carrying A broken jewelencrusted egg
Use compass directions for movement.
go around egg

Action: northwest
[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', '
    in', 'west'), ('west', 'west', 'north_of_house'), ('mailbox', 'is', 'animate'), ('
    mailbox', 'in', 'west'), ('you', 'is', 'you'), ('you', 'have', 'jewel'), ('you', 'have
    ', 'encrusted'), ('you', 'have', 'egg'), ('you', 'have', 'broken'), ('north_of_house',
     'north', 'west'), ('north_of_house', 'south', 'forest_path'), ('windows', 'in', '
    north_of_house'), ('windows', 'is', 'animate'), ('trees', 'in', 'north_of_house'), ('
    trees', 'is', 'animate'), ('trees', 'in', 'forest'), ('path', 'is', 'animate'), ('path
    ', 'in', 'north_of_house'), ('path', 'in', 'clearing'), ('north', 'is', 'animate'), ('
    north', 'in', 'north_of_house'), ('large', 'in', 'forest_path'), ('large', 'is', '
    animate'), ('large', 'in', 'forest'), ('forest_path', 'north', 'north_of_house'), ('
    forest_path', 'west', 'forest'), ('forest_path', 'south', 'clearing'), ('tree', 'in',
    'forest_path'), ('tree', 'is', 'animate'), ('south', 'is', 'animate'), ('south', 'in',
     'forest_path'), ('forest', 'is', 'animate'), ('forest', 'east', 'forest_path'), ('
    forest', 'in', 'forest'), ('forest', 'west', 'clearing'), ('mountains', 'is', 'animate
    '), ('mountains', 'in', 'forest'), ('clearing', 'north', 'forest_path'), ('pile', 'in
    ', 'clearing'), ('pile', 'is', 'animate'), ('leaves', 'is', 'animate'), ('leaves', 'in
    ', 'clearing'), ('east', 'is', 'animate'), ('east', 'in', 'forest'), ('egg', 'is', '
    animate'), ('nest', 'is', 'animate'), ('jewel', 'is', 'animate'), ('encrusted', 'is',
    'animate'), ('canary', 'is', 'animate'), ('canary', 'in', 'forest_path'), ('window', '
    is', 'animate'), ('broken', 'is', 'animate'), ('broken', 'in', 'forest_path'), ('
    clockwork', 'is', 'animate'), ('clockwork', 'in', 'forest_path'), ('clockwork_canary',
     'is', 'animate'), ('clockwork_canary', 'in', 'forest_path'), ('ground', 'is', '
    animate')]
==========STEP 40==========
Observation:
Forest This is a dimly lit forest, with large trees all around.
```

```
You are carrying A broken jewelencrusted egg
You cant go that way.
northwest

Action: south
[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', '
    in', 'west'), ('west', 'west', 'north_of_house'), ('mailbox', 'is', 'animate'), ('
    mailbox', 'in', 'west'), ('you', 'is', 'you'), ('you', 'have', 'jewel'), ('you', 'have
    ', 'encrusted'), ('you', 'have', 'egg'), ('you', 'have', 'broken'), ('north_of_house',
     'north', 'west'), ('north_of_house', 'south', 'forest_path'), ('windows', 'in', '
    north_of_house'), ('windows', 'is', 'animate'), ('trees', 'in', 'north_of_house'), ('
    trees', 'is', 'animate'), ('trees', 'in', 'forest'), ('path', 'is', 'animate'), ('path
    ', 'in', 'north_of_house'), ('path', 'in', 'clearing'), ('north', 'is', 'animate'), ('
    north', 'in', 'north_of_house'), ('large', 'in', 'forest_path'), ('large', 'is', '
    animate'), ('large', 'in', 'forest'), ('forest_path', 'north', 'north_of_house'), ('
    forest_path', 'west', 'forest'), ('forest_path', 'south', 'clearing'), ('tree', 'in',
    'forest_path'), ('tree', 'is', 'animate'), ('south', 'is', 'animate'), ('south', 'in',
     'forest_path'), ('forest', 'is', 'animate'), ('forest', 'east', 'forest_path'), ('
    forest', 'northwest', 'forest'), ('forest', 'west', 'clearing'), ('mountains', 'is', '
    animate'), ('mountains', 'in', 'forest'), ('clearing', 'north', 'forest_path'), ('pile
    ', 'in', 'clearing'), ('pile', 'is', 'animate'), ('leaves', 'is', 'animate'), ('leaves
    ', 'in', 'clearing'), ('east', 'is', 'animate'), ('east', 'in', 'forest'), ('egg', 'is
    ', 'animate'), ('nest', 'is', 'animate'), ('jewel', 'is', 'animate'), ('encrusted', '
    is', 'animate'), ('canary', 'is', 'animate'), ('canary', 'in', 'forest_path'), ('
    window', 'is', 'animate'), ('broken', 'is', 'animate'), ('broken', 'in', 'forest_path
    '), ('clockwork', 'is', 'animate'), ('clockwork', 'in', 'forest_path'), ('
    clockwork_canary', 'is', 'animate'), ('clockwork_canary', 'in', 'forest_path'), ('
    ground', 'is', 'animate')]
==========STEP 41==========
Observation:
Clearing You are in a small clearing in a well marked forest path that extends to the east
     and west.
You are carrying A broken jewelencrusted egg
Clearing You are in a small clearing in a well marked forest path that extends to the east
     and west.
south

Action: west
[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', '
    in', 'west'), ('west', 'west', 'north_of_house'), ('west', 'in', 'clearing'), ('
    mailbox', 'is', 'animate'), ('mailbox', 'in', 'west'), ('you', 'is', 'you'), ('you', '
```

have', 'jewel'), ('you', 'have', 'encrusted'), ('you', 'have', 'egg'), ('you', 'have', 'broken'), ('you', 'in', 'clearing'), ('north_of_house', 'north', 'west'), ('north_of_house', 'south', 'forest_path'), ('windows', 'in', 'north_of_house'), ('windows', 'is', 'animate'), ('trees', 'in', 'north_of_house'), ('trees', 'is', 'animate'), ('trees', 'in', 'forest'), ('path', 'is', 'animate'), ('path', 'in', 'north_of_house'), ('path', 'in', 'clearing'), ('north', 'is', 'animate'), ('north', 'in', 'north_of_house'), ('large', 'in', 'forest_path'), ('large', 'is', 'animate'), ('large', 'in', 'forest'), ('forest_path', 'north', 'north_of_house'), ('forest_path', 'west', 'forest'), ('forest_path', 'south', 'clearing'), ('tree', 'in', 'forest_path'), ('tree', 'is', 'animate'), ('south', 'is', 'animate'), ('south', 'in', 'forest_path'), ('forest', 'is', 'animate'), ('forest', 'east', 'forest_path'), ('forest', 'northwest', 'forest'), ('forest', 'in', 'clearing'), ('mountains', 'is', 'animate'), ('mountains', 'in', 'forest'), ('clearing', 'north', 'forest_path'), ('clearing', 'south', 'forest'), ('pile', 'in', 'clearing'), ('pile', 'is', 'animate'), ('leaves', 'is', 'animate'), ('leaves', 'in', 'clearing'), ('east', 'is', 'animate'), ('east', 'in', 'forest'), ('east', 'in', 'clearing'), ('egg', 'is', 'animate'), ('nest', 'is', 'animate'), ('jewel', 'is', 'animate'), ('encrusted', 'is', 'animate'), ('canary', 'is', 'animate'), ('canary', 'in', 'forest_path'), ('window', 'is', 'animate'), ('broken', 'is', 'animate'), ('broken', 'in', 'forest_path'), ('clockwork', 'is', 'animate'), ('clockwork', 'in', 'forest_path'), ('clockwork_canary', 'is', 'animate'), ('clockwork_canary', 'in', 'forest_path'), ('ground', 'is', 'animate')]

==========STEP 42==========

Observation:

Behind House You are behind the white house. A path leads into the forest to the east. In one corner of the house there is a small window which is slightly ajar.

You are carrying A broken jewelencrusted egg

Behind House You are behind the white house. A path leads into the forest to the east. In one corner of the house there is a small window which is slightly ajar.

west

Action: east

[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', 'in', 'west'), ('west', 'west', 'north_of_house'), ('west', 'in', 'clearing'), ('mailbox', 'is', 'animate'), ('mailbox', 'in', 'west'), ('you', 'is', 'you'), ('you', 'have', 'jewel'), ('you', 'have', 'encrusted'), ('you', 'have', 'egg'), ('you', 'have', 'broken'), ('you', 'in', 'behind_house'), ('north_of_house', 'north', 'west'), ('north_of_house', 'south', 'forest_path'), ('windows', 'in', 'north_of_house'), ('windows', 'is', 'animate'), ('trees', 'in', 'north_of_house'), ('trees', 'is', 'animate'), ('trees', 'in', 'forest'), ('path', 'is', 'animate'), ('path', 'in', 'north_of_house'), ('path', 'in', 'clearing'), ('north', 'is', 'animate'), ('north', 'in', 'north_of_house'), ('large', 'in', 'forest_path'), ('large', 'is', 'animate'), ('

large', 'in', 'forest'), ('forest_path', 'north', 'north_of_house'), ('forest_path', '
west', 'forest'), ('forest_path', 'south', 'clearing'), ('tree', 'in', 'forest_path'),
('tree', 'is', 'animate'), ('south', 'is', 'animate'), ('south', 'in', 'forest_path')
, ('forest', 'is', 'animate'), ('forest', 'east', 'forest_path'), ('forest', '
northwest', 'forest'), ('forest', 'in', 'clearing'), ('forest', 'in', 'behind_house'),
('mountains', 'is', 'animate'), ('mountains', 'in', 'forest'), ('clearing', 'north',
'forest_path'), ('clearing', 'south', 'forest'), ('pile', 'in', 'clearing'), ('pile',
'is', 'animate'), ('leaves', 'is', 'animate'), ('leaves', 'in', 'clearing'), ('east',
'is', 'animate'), ('east', 'in', 'forest'), ('east', 'in', 'clearing'), ('east', 'in',
'behind_house'), ('egg', 'is', 'animate'), ('nest', 'is', 'animate'), ('jewel', 'is',
'animate'), ('encrusted', 'is', 'animate'), ('canary', 'is', 'animate'), ('canary', '
in', 'forest_path'), ('window', 'is', 'animate'), ('window', 'in', 'behind_house'), ('
broken', 'is', 'animate'), ('broken', 'in', 'forest_path'), ('clockwork', 'is', '
animate'), ('clockwork', 'in', 'forest_path'), ('clockwork_canary', 'is', 'animate'),
('clockwork_canary', 'in', 'forest_path'), ('ground', 'is', 'animate'), ('behind_house
', 'west', 'clearing')]

==========STEP 43==========

Observation:

Clearing You are in a small clearing in a well marked forest path that extends to the east
and west.

You are carrying A broken jewelencrusted egg

Clearing

east

Action: shake egg

[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', '
in', 'west'), ('west', 'west', 'north_of_house'), ('west', 'in', 'clearing'), ('
mailbox', 'is', 'animate'), ('mailbox', 'in', 'west'), ('you', 'is', 'you'), ('you', '
have', 'jewel'), ('you', 'have', 'encrusted'), ('you', 'have', 'egg'), ('you', 'have',
'broken'), ('you', 'in', 'clearing'), ('north_of_house', 'north', 'west'), ('
north_of_house', 'south', 'forest_path'), ('windows', 'in', 'north_of_house'), ('
windows', 'is', 'animate'), ('trees', 'in', 'north_of_house'), ('trees', 'is', '
animate'), ('trees', 'in', 'forest'), ('path', 'is', 'animate'), ('path', 'in', '
north_of_house'), ('path', 'in', 'clearing'), ('north', 'is', 'animate'), ('north', '
in', 'north_of_house'), ('large', 'in', 'forest_path'), ('large', 'is', 'animate'), ('
large', 'in', 'forest'), ('forest_path', 'north', 'north_of_house'), ('forest_path', '
west', 'forest'), ('forest_path', 'south', 'clearing'), ('tree', 'in', 'forest_path'),
('tree', 'is', 'animate'), ('south', 'is', 'animate'), ('south', 'in', 'forest_path')
, ('forest', 'is', 'animate'), ('forest', 'east', 'forest_path'), ('forest', '
northwest', 'forest'), ('forest', 'in', 'clearing'), ('forest', 'in', 'behind_house'),
('mountains', 'is', 'animate'), ('mountains', 'in', 'forest'), ('clearing', 'north',

```
    'forest_path'), ('clearing', 'south', 'forest'), ('clearing', 'east', 'behind_house'),
     ('pile', 'in', 'clearing'), ('pile', 'is', 'animate'), ('leaves', 'is', 'animate'),
    ('leaves', 'in', 'clearing'), ('east', 'is', 'animate'), ('east', 'in', 'forest'), ('
    east', 'in', 'clearing'), ('east', 'in', 'behind_house'), ('egg', 'is', 'animate'), ('
    nest', 'is', 'animate'), ('jewel', 'is', 'animate'), ('encrusted', 'is', 'animate'),
    ('canary', 'is', 'animate'), ('canary', 'in', 'forest_path'), ('window', 'is', '
    animate'), ('window', 'in', 'behind_house'), ('broken', 'is', 'animate'), ('broken', '
    in', 'forest_path'), ('clockwork', 'is', 'animate'), ('clockwork', 'in', 'forest_path
    '), ('clockwork_canary', 'is', 'animate'), ('clockwork_canary', 'in', 'forest_path'),
    ('ground', 'is', 'animate'), ('behind_house', 'west', 'clearing')]
==========STEP 44==========
Observation:
Clearing You are in a small clearing in a well marked forest path that extends to the east
     and west.
You are carrying A broken jewelencrusted egg
Shaken.
shake egg

Action: south
[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', '
    in', 'west'), ('west', 'west', 'north_of_house'), ('west', 'in', 'clearing'), ('
    mailbox', 'is', 'animate'), ('mailbox', 'in', 'west'), ('you', 'is', 'you'), ('you', '
    have', 'jewel'), ('you', 'have', 'encrusted'), ('you', 'have', 'egg'), ('you', 'have',
     'broken'), ('north_of_house', 'north', 'west'), ('north_of_house', 'south', '
    forest_path'), ('windows', 'in', 'north_of_house'), ('windows', 'is', 'animate'), ('
    trees', 'in', 'north_of_house'), ('trees', 'is', 'animate'), ('trees', 'in', 'forest')
    , ('path', 'is', 'animate'), ('path', 'in', 'north_of_house'), ('path', 'in', '
    clearing'), ('north', 'is', 'animate'), ('north', 'in', 'north_of_house'), ('large', '
    in', 'forest_path'), ('large', 'is', 'animate'), ('large', 'in', 'forest'), ('
    forest_path', 'north', 'north_of_house'), ('forest_path', 'west', 'forest'), ('
    forest_path', 'south', 'clearing'), ('tree', 'in', 'forest_path'), ('tree', 'is', '
    animate'), ('south', 'is', 'animate'), ('south', 'in', 'forest_path'), ('forest', 'is
    ', 'animate'), ('forest', 'east', 'forest_path'), ('forest', 'northwest', 'forest'),
    ('forest', 'in', 'clearing'), ('forest', 'in', 'behind_house'), ('mountains', 'is', '
    animate'), ('mountains', 'in', 'forest'), ('clearing', 'north', 'forest_path'), ('
    clearing', 'south', 'forest'), ('clearing', 'east', 'behind_house'), ('pile', 'in', '
    clearing'), ('pile', 'is', 'animate'), ('leaves', 'is', 'animate'), ('leaves', 'in', '
    clearing'), ('east', 'is', 'animate'), ('east', 'in', 'forest'), ('east', 'in', '
    clearing'), ('east', 'in', 'behind_house'), ('egg', 'is', 'animate'), ('nest', 'is', '
    animate'), ('jewel', 'is', 'animate'), ('encrusted', 'is', 'animate'), ('canary', 'is
    ', 'animate'), ('canary', 'in', 'forest_path'), ('window', 'is', 'animate'), ('window
```

```
    ', 'in', 'behind_house'), ('broken', 'is', 'animate'), ('broken', 'in', 'forest_path')
    , ('clockwork', 'is', 'animate'), ('clockwork', 'in', 'forest_path'), ('
    clockwork_canary', 'is', 'animate'), ('clockwork_canary', 'in', 'forest_path'), ('
    ground', 'is', 'animate'), ('behind_house', 'west', 'clearing')]
==========STEP 45==========
Observation:
Forest This is a dimly lit forest, with large trees all around.
You are carrying A broken jewelencrusted egg
Forest This is a dimly lit forest, with large trees all around.
south

Action: go around egg
[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', '
    in', 'west'), ('west', 'west', 'north_of_house'), ('west', 'in', 'clearing'), ('
    mailbox', 'is', 'animate'), ('mailbox', 'in', 'west'), ('you', 'is', 'you'), ('you', '
    have', 'jewel'), ('you', 'have', 'encrusted'), ('you', 'have', 'egg'), ('you', 'have',
     'broken'), ('you', 'in', 'forest'), ('north_of_house', 'north', 'west'), ('
    north_of_house', 'south', 'forest_path'), ('windows', 'in', 'north_of_house'), ('
    windows', 'is', 'animate'), ('trees', 'in', 'north_of_house'), ('trees', 'is', '
    animate'), ('trees', 'in', 'forest'), ('path', 'is', 'animate'), ('path', 'in', '
    north_of_house'), ('path', 'in', 'clearing'), ('north', 'is', 'animate'), ('north', '
    in', 'north_of_house'), ('large', 'in', 'forest_path'), ('large', 'is', 'animate'), ('
    large', 'in', 'forest'), ('forest_path', 'north', 'north_of_house'), ('forest_path', '
    west', 'forest'), ('forest_path', 'south', 'clearing'), ('tree', 'in', 'forest_path'),
     ('tree', 'is', 'animate'), ('south', 'is', 'animate'), ('south', 'in', 'forest_path')
    , ('forest', 'is', 'animate'), ('forest', 'east', 'forest_path'), ('forest', 'in', '
    forest'), ('forest', 'south', 'clearing'), ('forest', 'in', 'behind_house'), ('
    mountains', 'is', 'animate'), ('mountains', 'in', 'forest'), ('clearing', 'north', '
    forest_path'), ('clearing', 'south', 'forest'), ('clearing', 'east', 'behind_house'),
    ('pile', 'in', 'clearing'), ('pile', 'is', 'animate'), ('leaves', 'is', 'animate'), ('
    leaves', 'in', 'clearing'), ('east', 'is', 'animate'), ('east', 'in', 'forest'), ('
    east', 'in', 'clearing'), ('east', 'in', 'behind_house'), ('egg', 'is', 'animate'), ('
    nest', 'is', 'animate'), ('jewel', 'is', 'animate'), ('encrusted', 'is', 'animate'),
    ('canary', 'is', 'animate'), ('canary', 'in', 'forest_path'), ('window', 'is', '
    animate'), ('window', 'in', 'behind_house'), ('broken', 'is', 'animate'), ('broken', '
    in', 'forest_path'), ('clockwork', 'is', 'animate'), ('clockwork', 'in', 'forest_path
    '), ('clockwork_canary', 'is', 'animate'), ('clockwork_canary', 'in', 'forest_path'),
    ('ground', 'is', 'animate'), ('behind_house', 'west', 'clearing')]
==========STEP 46==========
Observation:
Forest This is a dimly lit forest, with large trees all around.
```

```
You are carrying A broken jewelencrusted egg
Use compass directions for movement.
go around egg

Action: go around egg
[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', '
    in', 'west'), ('west', 'west', 'north_of_house'), ('west', 'in', 'clearing'), ('
    mailbox', 'is', 'animate'), ('mailbox', 'in', 'west'), ('you', 'is', 'you'), ('you', '
    have', 'jewel'), ('you', 'have', 'encrusted'), ('you', 'have', 'egg'), ('you', 'have',
     'broken'), ('north_of_house', 'north', 'west'), ('north_of_house', 'south', '
    forest_path'), ('windows', 'in', 'north_of_house'), ('windows', 'is', 'animate'), ('
    trees', 'in', 'north_of_house'), ('trees', 'is', 'animate'), ('trees', 'in', 'forest')
    , ('path', 'is', 'animate'), ('path', 'in', 'north_of_house'), ('path', 'in', '
    clearing'), ('north', 'is', 'animate'), ('north', 'in', 'north_of_house'), ('large', '
    in', 'forest_path'), ('large', 'is', 'animate'), ('large', 'in', 'forest'), ('
    forest_path', 'north', 'north_of_house'), ('forest_path', 'west', 'forest'), ('
    forest_path', 'south', 'clearing'), ('tree', 'in', 'forest_path'), ('tree', 'is', '
    animate'), ('south', 'is', 'animate'), ('south', 'in', 'forest_path'), ('forest', 'is
    ', 'animate'), ('forest', 'east', 'forest_path'), ('forest', 'in', 'forest'), ('forest
    ', 'south', 'clearing'), ('forest', 'in', 'behind_house'), ('mountains', 'is', '
    animate'), ('mountains', 'in', 'forest'), ('clearing', 'north', 'forest_path'), ('
    clearing', 'south', 'forest'), ('clearing', 'east', 'behind_house'), ('pile', 'in', '
    clearing'), ('pile', 'is', 'animate'), ('leaves', 'is', 'animate'), ('leaves', 'in', '
    clearing'), ('east', 'is', 'animate'), ('east', 'in', 'forest'), ('east', 'in', '
    clearing'), ('east', 'in', 'behind_house'), ('egg', 'is', 'animate'), ('nest', 'is', '
    animate'), ('jewel', 'is', 'animate'), ('encrusted', 'is', 'animate'), ('canary', 'is
    ', 'animate'), ('canary', 'in', 'forest_path'), ('window', 'is', 'animate'), ('window
    ', 'in', 'behind_house'), ('broken', 'is', 'animate'), ('broken', 'in', 'forest_path')
    , ('clockwork', 'is', 'animate'), ('clockwork', 'in', 'forest_path'), ('
    clockwork_canary', 'is', 'animate'), ('clockwork_canary', 'in', 'forest_path'), ('
    ground', 'is', 'animate'), ('behind_house', 'west', 'clearing')]
==========STEP 47==========
Observation:
Forest This is a dimly lit forest, with large trees all around.
You are carrying A broken jewelencrusted egg
Use compass directions for movement.
go around egg

Action: west
[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', '
    in', 'west'), ('west', 'west', 'north_of_house'), ('west', 'in', 'clearing'), ('
```

mailbox', 'is', 'animate'), ('mailbox', 'in', 'west'), ('you', 'is', 'you'), ('you', '
have', 'jewel'), ('you', 'have', 'encrusted'), ('you', 'have', 'egg'), ('you', 'have',
'broken'), ('north_of_house', 'north', 'west'), ('north_of_house', 'south', '
forest_path'), ('windows', 'in', 'north_of_house'), ('windows', 'is', 'animate'), ('
trees', 'in', 'north_of_house'), ('trees', 'is', 'animate'), ('trees', 'in', 'forest')
, ('path', 'is', 'animate'), ('path', 'in', 'north_of_house'), ('path', 'in', '
clearing'), ('north', 'is', 'animate'), ('north', 'in', 'north_of_house'), ('large', '
in', 'forest_path'), ('large', 'is', 'animate'), ('large', 'in', 'forest'), ('
forest_path', 'north', 'north_of_house'), ('forest_path', 'west', 'forest'), ('
forest_path', 'south', 'clearing'), ('tree', 'in', 'forest_path'), ('tree', 'is', '
animate'), ('south', 'is', 'animate'), ('south', 'in', 'forest_path'), ('forest', 'is
', 'animate'), ('forest', 'east', 'forest_path'), ('forest', 'in', 'forest'), ('forest
', 'south', 'clearing'), ('forest', 'in', 'behind_house'), ('mountains', 'is', '
animate'), ('mountains', 'in', 'forest'), ('clearing', 'north', 'forest_path'), ('
clearing', 'south', 'forest'), ('clearing', 'east', 'behind_house'), ('pile', 'in', '
clearing'), ('pile', 'is', 'animate'), ('leaves', 'is', 'animate'), ('leaves', 'in', '
clearing'), ('east', 'is', 'animate'), ('east', 'in', 'forest'), ('east', 'in', '
clearing'), ('east', 'in', 'behind_house'), ('egg', 'is', 'animate'), ('nest', 'is', '
animate'), ('jewel', 'is', 'animate'), ('encrusted', 'is', 'animate'), ('canary', 'is
', 'animate'), ('canary', 'in', 'forest_path'), ('window', 'is', 'animate'), ('window
', 'in', 'behind_house'), ('broken', 'is', 'animate'), ('broken', 'in', 'forest_path')
, ('clockwork', 'is', 'animate'), ('clockwork', 'in', 'forest_path'), ('
clockwork_canary', 'is', 'animate'), ('clockwork_canary', 'in', 'forest_path'), ('
ground', 'is', 'animate'), ('behind_house', 'west', 'clearing')]

==========STEP 48==========

Observation:

Forest This is a forest, with trees in all directions. To the east, there appears to be
sunlight.

You are carrying A broken jewelencrusted egg

Forest

west

Action: south

[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', '
in', 'west'), ('west', 'west', 'north_of_house'), ('west', 'in', 'clearing'), ('
mailbox', 'is', 'animate'), ('mailbox', 'in', 'west'), ('you', 'is', 'you'), ('you', '
have', 'jewel'), ('you', 'have', 'encrusted'), ('you', 'have', 'egg'), ('you', 'have',
'broken'), ('north_of_house', 'north', 'west'), ('north_of_house', 'south', '
forest_path'), ('windows', 'in', 'north_of_house'), ('windows', 'is', 'animate'), ('
trees', 'in', 'north_of_house'), ('trees', 'is', 'animate'), ('trees', 'in', 'forest')
, ('path', 'is', 'animate'), ('path', 'in', 'north_of_house'), ('path', 'in', '

clearing'), ('north', 'is', 'animate'), ('north', 'in', 'north_of_house'), ('large', '
in', 'forest_path'), ('large', 'is', 'animate'), ('large', 'in', 'forest'), ('
forest_path', 'north', 'north_of_house'), ('forest_path', 'west', 'forest'), ('
forest_path', 'south', 'clearing'), ('tree', 'in', 'forest_path'), ('tree', 'is', '
animate'), ('south', 'is', 'animate'), ('south', 'in', 'forest_path'), ('forest', 'is
', 'animate'), ('forest', 'east', 'forest_path'), ('forest', 'west', 'forest'), ('
forest', 'south', 'clearing'), ('forest', 'in', 'behind_house'), ('mountains', 'is', '
animate'), ('mountains', 'in', 'forest'), ('clearing', 'north', 'forest_path'), ('
clearing', 'south', 'forest'), ('clearing', 'east', 'behind_house'), ('pile', 'in', '
clearing'), ('pile', 'is', 'animate'), ('leaves', 'is', 'animate'), ('leaves', 'in', '
clearing'), ('east', 'is', 'animate'), ('east', 'in', 'forest'), ('east', 'in', '
clearing'), ('east', 'in', 'behind_house'), ('egg', 'is', 'animate'), ('nest', 'is', '
animate'), ('jewel', 'is', 'animate'), ('encrusted', 'is', 'animate'), ('canary', 'is
', 'animate'), ('canary', 'in', 'forest_path'), ('window', 'is', 'animate'), ('window
', 'in', 'behind_house'), ('broken', 'is', 'animate'), ('broken', 'in', 'forest_path')
, ('clockwork', 'is', 'animate'), ('clockwork', 'in', 'forest_path'), ('
clockwork_canary', 'is', 'animate'), ('clockwork_canary', 'in', 'forest_path'), ('
ground', 'is', 'animate'), ('behind_house', 'west', 'clearing')]

==========STEP 49==========

Observation:

Forest This is a dimly lit forest, with large trees all around.

You are carrying A broken jewelencrusted egg

Forest

south

Action: throw forest at forest

[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', '
in', 'west'), ('west', 'west', 'north_of_house'), ('west', 'in', 'clearing'), ('
mailbox', 'is', 'animate'), ('mailbox', 'in', 'west'), ('you', 'is', 'you'), ('you', '
have', 'jewel'), ('you', 'have', 'encrusted'), ('you', 'have', 'egg'), ('you', 'have',
 'broken'), ('north_of_house', 'north', 'west'), ('north_of_house', 'south', '
forest_path'), ('windows', 'in', 'north_of_house'), ('windows', 'is', 'animate'), ('
trees', 'in', 'north_of_house'), ('trees', 'is', 'animate'), ('trees', 'in', 'forest')
, ('path', 'is', 'animate'), ('path', 'in', 'north_of_house'), ('path', 'in', '
clearing'), ('north', 'is', 'animate'), ('north', 'in', 'north_of_house'), ('large', '
in', 'forest_path'), ('large', 'is', 'animate'), ('large', 'in', 'forest'), ('
forest_path', 'north', 'north_of_house'), ('forest_path', 'west', 'forest'), ('
forest_path', 'south', 'clearing'), ('tree', 'in', 'forest_path'), ('tree', 'is', '
animate'), ('south', 'is', 'animate'), ('south', 'in', 'forest_path'), ('forest', 'is
', 'animate'), ('forest', 'east', 'forest_path'), ('forest', 'in', 'forest'), ('forest
', 'south', 'clearing'), ('forest', 'in', 'behind_house'), ('mountains', 'is', '

```
    animate'), ('mountains', 'in', 'forest'), ('clearing', 'north', 'forest_path'), ('
    clearing', 'south', 'forest'), ('clearing', 'east', 'behind_house'), ('pile', 'in', '
    clearing'), ('pile', 'is', 'animate'), ('leaves', 'is', 'animate'), ('leaves', 'in', '
    clearing'), ('east', 'is', 'animate'), ('east', 'in', 'forest'), ('east', 'in', '
    clearing'), ('east', 'in', 'behind_house'), ('egg', 'is', 'animate'), ('nest', 'is', '
    animate'), ('jewel', 'is', 'animate'), ('encrusted', 'is', 'animate'), ('canary', 'is
    ', 'animate'), ('canary', 'in', 'forest_path'), ('window', 'is', 'animate'), ('window
    ', 'in', 'behind_house'), ('broken', 'is', 'animate'), ('broken', 'in', 'forest_path')
    , ('clockwork', 'is', 'animate'), ('clockwork', 'in', 'forest_path'), ('
    clockwork_canary', 'is', 'animate'), ('clockwork_canary', 'in', 'forest_path'), ('
    ground', 'is', 'animate'), ('behind_house', 'west', 'clearing')]
==========STEP 50==========
Observation:
Forest This is a dimly lit forest, with large trees all around.
You are carrying A broken jewelencrusted egg
You dont have the forest.
throw forest at forest

Action: north
[('door', 'in', 'west'), ('door', 'is', 'animate'), ('west', 'is', 'animate'), ('west', '
    in', 'west'), ('west', 'west', 'north_of_house'), ('west', 'in', 'clearing'), ('
    mailbox', 'is', 'animate'), ('mailbox', 'in', 'west'), ('you', 'is', 'you'), ('you', '
    have', 'jewel'), ('you', 'have', 'encrusted'), ('you', 'have', 'egg'), ('you', 'have',
     'broken'), ('north_of_house', 'north', 'west'), ('north_of_house', 'south', '
    forest_path'), ('windows', 'in', 'north_of_house'), ('windows', 'is', 'animate'), ('
    trees', 'in', 'north_of_house'), ('trees', 'is', 'animate'), ('trees', 'in', 'forest')
    , ('path', 'is', 'animate'), ('path', 'in', 'north_of_house'), ('path', 'in', '
    clearing'), ('north', 'is', 'animate'), ('north', 'in', 'north_of_house'), ('large', '
    in', 'forest_path'), ('large', 'is', 'animate'), ('large', 'in', 'forest'), ('
    forest_path', 'north', 'north_of_house'), ('forest_path', 'west', 'forest'), ('
    forest_path', 'south', 'clearing'), ('tree', 'in', 'forest_path'), ('tree', 'is', '
    animate'), ('south', 'is', 'animate'), ('south', 'in', 'forest_path'), ('forest', 'is
    ', 'animate'), ('forest', 'east', 'forest_path'), ('forest', 'in', 'forest'), ('forest
    ', 'south', 'clearing'), ('forest', 'in', 'behind_house'), ('mountains', 'is', '
    animate'), ('mountains', 'in', 'forest'), ('clearing', 'north', 'forest_path'), ('
    clearing', 'south', 'forest'), ('clearing', 'east', 'behind_house'), ('pile', 'in', '
    clearing'), ('pile', 'is', 'animate'), ('leaves', 'is', 'animate'), ('leaves', 'in', '
    clearing'), ('east', 'is', 'animate'), ('east', 'in', 'forest'), ('east', 'in', '
    clearing'), ('east', 'in', 'behind_house'), ('egg', 'is', 'animate'), ('nest', 'is', '
    animate'), ('jewel', 'is', 'animate'), ('encrusted', 'is', 'animate'), ('canary', 'is
    ', 'animate'), ('canary', 'in', 'forest_path'), ('window', 'is', 'animate'), ('window
```

```
', 'in', 'behind_house'), ('broken', 'is', 'animate'), ('broken', 'in', 'forest_path')
, ('clockwork', 'is', 'animate'), ('clockwork', 'in', 'forest_path'), ('
clockwork_canary', 'is', 'animate'), ('clockwork_canary', 'in', 'forest_path'), ('
ground', 'is', 'animate'), ('behind_house', 'west', 'clearing')]
```


A.4 Datasheet

I provide comprehensive documentation of the dataset based on Datasheets for Datasets (Gebru *et al.* 2018).

A.4.1 Motivation

For what purpose was the dataset created? Was there a specific task in mind? Was there a specific gap that needed to be filled? Please provide a description. I seek to create agents that exhibit human-like capabilities such as commonsense reasoning and natural language understanding in interactive and situated settings. In pursuit of this goal, I provide a dataset that enables the creation of learning agents that can build knowledge graph-based world models of interactive narratives.

Who created this dataset (e.g., which team, research group) and on behalf of which entity (e.g., company, institution, organization)? It was created by Prithviraj Ammanabrolu and Mark Riedl at the Georgia Institute of Technology.

Who funded the creation of the dataset? If there is an associated grant, please provide the name of the grantor and the grant name and number. It was funded by the US's Defense Advanced Research Projects Agency (DARPA) as part of a fundamental science research grant Science of Artificial Intelligence and Learning for Open-world Novelty (SAIL-ON https://www.darpa.mil/program/science-of-artificial-intelligence-and-learning-for-open-world-novelty).

A.4.2 Composition

What do the instances that comprise the dataset represent (e.g., documents, photos, people, countries)? Are there multiple types of instances (e.g., movies, users, and ratings; people and interactions between them; nodes and edges)? Please provide a description. Each instance of our dataset takes the tuples of $\langle s_t, a_t, s_{t+1}, r_{t+1} \rangle$ where s_t and s_{t+1} are two subsequent states of a text game with a_t being the action used to transition states and r_{t+1} is the observed reward for some step t. Everything is in text. These are all collected from various text games and examples of instances are found in Appendix A.1.

How many instances are there in total (of each type, if appropriate)? The training data has 24198 mappings and is collected across 27 games in multiple genres and contains a further 7836 heldout instances over 9 additional games in the test set.

Does the dataset contain all possible instances or is it a sample (not necessarily random) of instances from a larger set? If the dataset is a sample, then what is the larger set? Is the sample representative of the larger set (e.g., geographic coverage)? If so, please describe how this representativeness was validated/verified. If it is not representative of the larger set, please describe why not (e.g., to cover a more diverse range of instances, because instances were withheld or unavailable). The dataset is a sample of the larger set of all possible states in each game. The samples are made to be biased towards states near the walkthroughs required to finish a game.

What data does each instance consist of? "Raw" data (e.g., unprocessed text or images)or features? In either case, please provide a description. Data is all in the form of text, either raw or in structured knowledge graph form.

Is there a label or target associated with each instance? If so, please provide a description. The data has multiple fields, depending on the tasks defined any of them can be used as labels. E.g. the knowledge graph prediction task has the graph field as the target.

Is any information missing from individual instances? If so, please provide a description, explaining why this information is missing (e.g., because it was unavail-

able). This does not include intentionally removed information, but might include, e.g., redacted text Not all games have human readable attributes for objects—when they do not, these are omitted by leaving the attributes fields blank. All other data is present for all instances.

Are relationships between individual instances made explicit (e.g., users' movie ratings, social network links)? If so, please describe how these relationships are made explicit. Instances are grouped together by game through the game field.

Are there recommended data splits (e.g., training, development/validation, testing)? If so, please provide a description of these splits, explaining the rationale behind them. I provide a training split of 27 games, and a testing split of 9 games. These are selected on the basis of existing works and each split contains a diverse set of games in terms of genre.

Are there any errors, sources of noise, or redundancies in the dataset? If so, please provide a description.

Is the dataset self-contained, or does it link to or otherwise rely on external resources (e.g., websites, tweets, other datasets)? If it links to or relies on external resources, a) are there guarantees that they will exist, and remain constant, over time; b) are there official archival versions of the complete dataset (i.e., including the external resources as they existed at the time the dataset was created); c) are there any restrictions (e.g., licenses, fees) associated with any of the external resources that might apply to a future user? Please provide descriptions of all external resources and any restrictions associated with them, as well as links or other access points, as appropriate. The creation of the dataset depends on the Jericho framework https://github.com/microsoft/jericho but the archival versions themselves do not have any dependencies.

Does the dataset contain data that might be considered confidentiality, data that includes the content of individuals non-public communications)? If so, please provide

a description. No, all data is part of games that are already public.

Does the dataset contain data that, if viewed directly, might be offensive, insulting, threatening, or might otherwise cause anxiety? If so, please describe why. The data is collected from games containing situations of non-normative language usage—describing situations that fictional characters may engage in that are potentially inappropriate, and on occasion impossible, for the real world such as running a troll through with a sword. Instances of such scenarios are mitigated by careful curation of the games that the data is collected from. The original Jericho framework (Hausknecht *et al.* 2020)—further verified by us in this work—uses a curated set of games found not to contain extreme examples of non-normative language usage. This is based on manual vetting and (existing) crowd-sourced reviews on the popular interactive narrative forum IFDB https://ifdb.org/.

A.4.3 Collection

How was the data associated with each instance acquired? Was the data directly observable (e.g., raw text, movie ratings), reported by subjects (e.g., survey responses), or indirectly inferred/derived from other data (e.g., part-of-speech tags, model-based guesses for age or language)? If data was reported by subjects or indirectly inferred/derived from other data, was the data validated/verified? If so, please describe how I build off the popular text game simulator Jericho (Hausknecht *et al.* 2020), I have constructed a dataset dubbed Worldformer that maps text game state observations to both the underlying ground truth knowledge graph representations of the game and the set of contextually relevant actions that can be performed in that state.

What mechanisms or procedures were used to collect the data (e.g., hardware apparatus or sensor, manual human curation, software program, software API)? How were these mechanisms or procedures validated? To collect the $\langle s_t, a_t, s_{t+1}, r_{t+1} \rangle$ tuples I implement a basic agent that explores the game along a trajectory corresponding to a *game walkthrough*. Game walkthroughs are texts describing the solutions to games, gener-

ally retrieved from the internet, but already part of the Jericho framework. Walkthroughs, however, only present one possible solution to a game and solve all the core puzzles required to complete a game with the maximum possible score. To achieve greater coverage of the game's state space, our data collection agent stops off to explore by executing random valid actions for n steps before resetting to the walkthrough.

If the dataset is a sample from a larger set, what was the sampling strategy (e.g., deterministic, probabilistic with specific sampling probabilities)? Randomly sampled actions are based on a random seed in Python's random package https://docs.python.org/3/library/random.html. I provide a seed and the specific package version.

Who was involved in the data collection process (e.g., students, crowdworkers, contractors) and how were they compensated (e.g., how much were crowdworkers paid)? Only the authors were involved, building on the contributions of the Jericho developers.

Over what timeframe was the data collected? Does this timeframe match the creation timeframe of the data associated with the instances (e.g., recent crawl of old news articles)? If not, please describe the timeframe in which the data associated with the instances was created. This dataset was developed over a period of 6 months, though the games used within date back to the 1970s.

Were any ethical review processes conducted (e.g., by an institutional review board)? If so, please provide a description of these review processes, including the outcomes, as well as a link or other access point to any supporting documentation. No human subjects were involved, no IRB process was undertaken.

A.4.4 Preprocessing

Was any preprocessing/cleaning/labeling of the data done (e.g., discretization or bucketing, tokenization, part-of-speech tagging, SIFT feature extraction, removal of instances, processing of missing values)? If so, please provide a description. If not, you may skip the remainder of the questions in this section. Games were decompiled to

extract attributes and ground truth knowledge graphs, the creation script is provided in the GitHub repo.

Was the "raw" data saved in addition to the preprocessed/cleaned/labeled data (e.g., to support unanticipated future uses)? If so, please provide a link or other access point to the "raw" data. No, raw binary game states were not saved and were converted to human readable text.

Is the software used to preprocess/clean/label the instances available? If so, please provide a link or other access point. Games were decompiled to extract attributes and ground truth knowledge graphs, the creation script will be provided in the GitHub repository.

A.4.5 Uses

Has the dataset been used for any tasks already? If so, please provide a description. No.

Is there a repository that links to any or all papers or systems that use the dataset? If so, please provide a link or other access point. No.

What (other) tasks could the dataset be used for? There are many more tasks that can be framed for other challenges related to world modeling from this dataset. Some immediate examples: (1) offline reinforcement learning for game agents through imitation learning—predicting the sequence of actions that finish the game based on walkthroughs and reward information; (2) knowledge graph verbalization, a form of the standard data-to-text natural language processing task (Wiseman *et al.* 2017), in which I learn to generate text that is conditioned on a knowledge graph; and (3) description generation conditioned on the names of various objects, locations, and characters—with applications in long-form text generation domains such as automated storytelling (Fan *et al.* 2019; Martin *et al.* 2018) and procedural generation of interactive narratives (Ammanabrolu *et al.* 2020b; Walton *et al.* 2020).

Is there anything about the composition of the dataset or the way it was collected and preprocessed/cleaned/labeled that might impact future uses? For example, is there anything that a future user might need to know to avoid uses that could result in unfair treatment of individuals or groups (e.g., stereotyping, quality of service issues) or other undesirable harms (e.g., financial harms, legal risks) If so, please provide a description. Is there anything a future user could do to mitigate these undesirable harms? Users should keep in mind that these come from games and can potentially describe non-normative situations.

Are there tasks for which the dataset should not be used? If so, please provide a description This dataset should not be used for tasks that involve direct physical interactions with humans, such as robotics.

A.4.6 Distribution

Will the dataset be distributed to third parties outside of the entity (e.g., company, institution, organization) on behalf of which the dataset was created? If so, please provide a description. It is open-sourced.

How will the dataset will be distributed (e.g., tarball on website, API, GitHub)? Does the dataset have a digital object identifier (DOI)? The dataset will be open-sourced at https://github.com/JerichoWorld/JerichoWorld.

When will the dataset be distributed? It was first released in May 2021.

Will the dataset be distributed under a copyright or other intellectual property (IP) license, and/or under applicable terms of use (ToU)? If so, please describe this license and/or ToU, and provide a link or other access point to, or otherwise reproduce, any relevant licensing terms or ToU, as well as any fees associated with these restrictions. The dataset will be under an MIT license, this is indicated on the GitHub repository.

Have any third parties imposed IP-based or other restrictions on the data associ-

ated with the instances? If so, please describe these restrictions, and provide a link or other access point to, or otherwise reproduce, any relevant licensing terms, as well as any fees associated with these restrictions. No.

Do any export controls or other regulatory restrictions apply to the dataset or to individual instances? If so, please describe these restrictions, and provide a link or other access point to, or otherwise reproduce, any supporting documentation. No.

A.4.7 Maintenance

Who is supporting/hosting/maintaining the dataset? Prithviraj Ammanabrolu will be responsible for maintenance.

How can the owner/curator/manager of the dataset be contacted (e.g., email address)? `raj.ammanabrolu@gatech.edu` or by filing an issue on the GitHub.

Is there an erratum? If so, please provide a link or other access point No.

Will the dataset be updated (e.g., to correct labeling errors, add new instances, delete instances)? If so, please describe how often, by whom, and how updates will be communicated to users (e.g., mailing list, GitHub)? Yes, more games will be added and corresponding data will be collected. Previous versions will be kept for backwards compatibility.

If the dataset relates to people, are there applicable limits on the retention of the data associated with the instances (e.g., were individuals in question told that their data would be retained for a fixed period of time and then deleted)? If so, please describe these limits and explain how they will be enforced. No.

Will older versions of the dataset continue to be supported/hosted/maintained? If so, please describe how. If not, please describe how its obsolescence will be communicated to users. Yes, versions will be archived on the GitHub repository.

If others want to extend/augment/build on/contribute to the dataset, is there a mechanism for them to do so? If so, please provide a description. Will these contri-

butions be validated/verified? If so, please describe how. If not, why not? Is there a process for communicating/distributing these contributions to other users? If so, please provide a description They can fork and submit pull requests to the current repository if they wish to extend it—these will be validated in an open-source manner on GitHub via reviews of the extensions.

APPENDIX B

GAME PLAYING EXPERIMENTS

B.1 KG-A2C

Episodes are terminated after 100 valid steps or game over/victory. Agents that decode invalid actions often wouldn't make it very far into the game, and so I only count valid-actions against the hundred step limit. All agents are trained individually on each game and then evaluated on that game. All A2C based agents are trained using data collected from 32 parallel environments. TDQN was trained using a single environment. Hyperparameters for all agents were tuned on the game of *Zork1* and held constant across all other games. Final reported scores are an average over 5 runs of each algorithm.

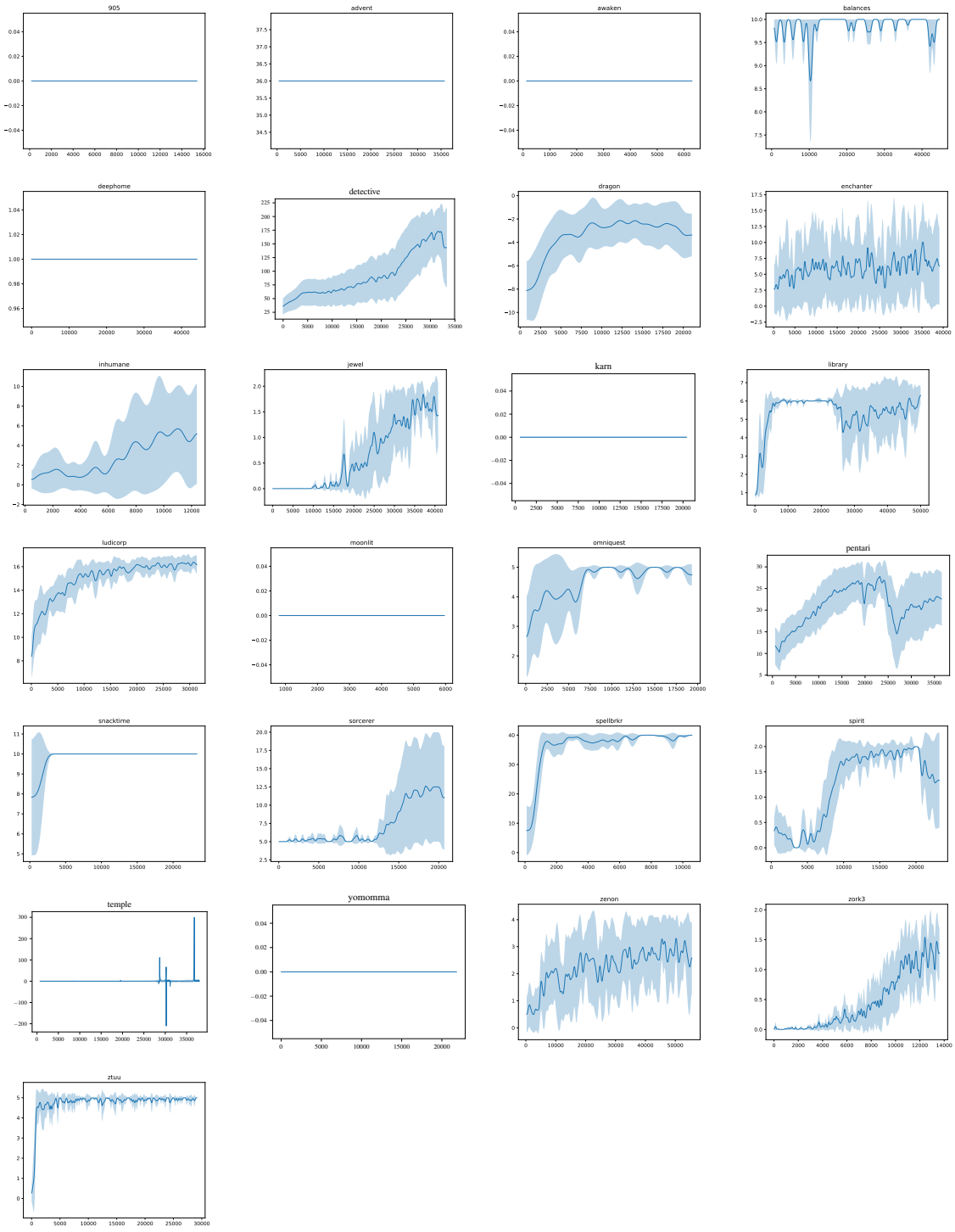


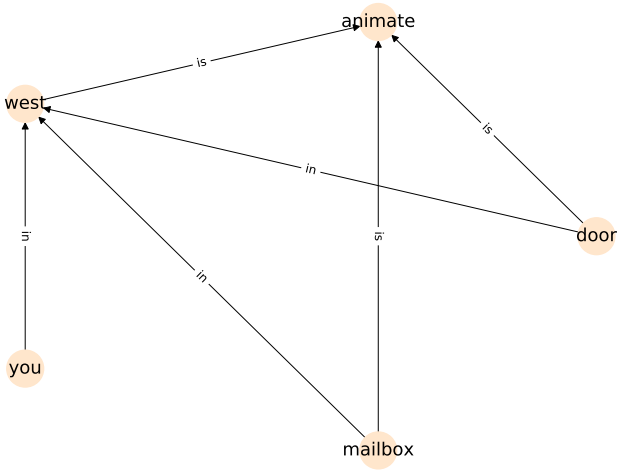
Figure B.1: Learning curves for KGA2C-full. Shaded regions indicate standard deviations.

B.2 Q*BERT

This section outlines how Q*BERT knowledge graph examples over a particular trajectory, architecture details, MC!Q*BERT/GO!Q*BERT training details, and hyperparameter details for all experiments.

B.2.1 Q*BERT Knowledge Graph Update Examples

Below is an excerpt from *Zork1* showing the exact observations given to the Q*BERT,the knowledge graph, and the corresponding action taken by the agent after the graph extraction and update process has occurred as described above for a trajectory consisting of 5 timesteps. These timesteps begin at the start of the game in *West of House* and continue till the agent has entered the *Kitchen* as seen in Fig. B.4. The set of $\langle s, r, o \rangle$ triples that make up the graph are in the text and the figure shows a partial visualization of the graph at that particular step in the trajectory.



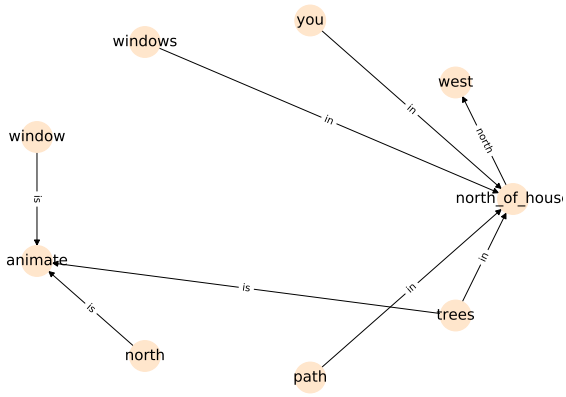
[loc] West of House You are standing in an open field west of a white house, with a boarded front door. There is a small mailbox here. [inv] You are empty handed.

[obs] Copyright c 1981, 1982, 1983 Infocom, Inc. All rights reserved. ZORK is a registered trademark of Infocom, Inc. Revision 88 / Serial number 840726 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 here.

[atr] talkable, seen, lieable, enterable, nodwarf, indoors, visited, handed, lockable, surface, thing, water_room, unlock, lost, afflicted, is_treasure, converse, mentioned, male, npcworn, no_article, relevant, scored, queryable, town, pluggable, happy, is_followable, legible, multitude, burning, room, clothing, underneath, ward_area, little, intact, animate, bled_in, supporter, readable, openable, near, nonlocal, door, plugged, sittable, toolbit, vehicle, light, lens_searchable, open, familiar, is_scroll, aimable, takeable, static, unique, concealed, vowelstart, alcoholic, bodypart, general, is_spell, full, dry_land, pushable, known, proper, inside, clean, ambiguously_plural, container, edible, treasure, can_plug, weapon, is_arrow, insubstantial, pluralname, transparent, is_coin, air_room, scenery, on, is_spell_book, burnt, burnable, auto_searched, locked, switchable, absent, rockable, beenunlocked, progressing, severed, worn, windy, stone, random, neuter, legible, female, asleep, wiped

[graph] [(you, in, west), (door, is, animate), (door, in, west), (west, is, animate), (mailbox, in, west), (mailbox, is, animate)]

[next act] go north



[loc] North of House You are facing the north side of a white house. There is no door here , and all the windows are boarded up. To the north a narrow path winds through the trees.

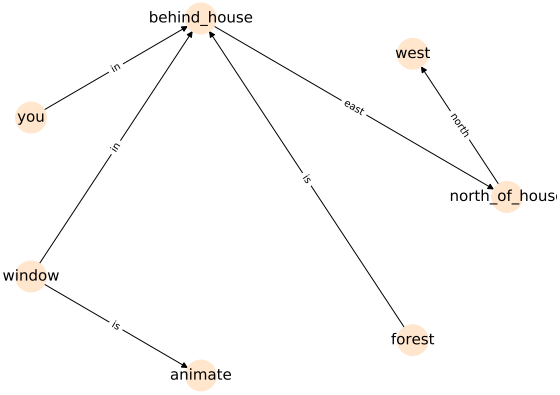
[inv] You are empty handed.

[obs] North of House You are facing the north side of a white house. There is no door here , and all the windows are boarded up. To the north a narrow path winds through the trees.

[atr] talkable, seen, lieable, enterable, nodwarf, indoors, visited, handed, lockable, surface, thing, water_room, unlock, lost, afflicted, is_treasure, converse, mentioned, male, npcworn, no_article, relevant, scored, queryable, town, pluggable, happy, is_followable, legible, multitude, burning, room, clothing, underneath, ward_area, little, intact, animate, bled_in, supporter, readable, openable, near, nonlocal, door, plugged, sittable, toolbit, vehicle, light, lens_searchable, open, familiar, is_scroll, aimable, takeable, static, unique, concealed, vowelstart, alcoholic, bodypart, general, is_spell, full, dry_land, pushable, known, proper, inside, clean, ambiguously_plural, container, edible, treasure, can_plug, weapon, is_arrow, insubstantial, pluralname, transparent, is_coin, air_room, scenery, on, is_spell_book, burnt, burnable, auto_searched, locked, switchable, absent, rockable, beenunlocked, progressing, severed, worn, windy, stone, random, neuter, legible, female, asleep, wiped

[graph] [(north_of_house, north, west), (you, in, north_of_house), (door, is, animate), (door, in, west), (west, is, animate), (west, in, west), (mailbox, in, west), (mailbox, is, animate), (windows, in, north_of_house), (windows, is, animate), (north, is, animate), (north, in, north_of_house), (path, is, animate), (path, in, north_of_house), (trees, in, north_of_house), (trees, is, animate)]

[next act] go east



[loc] Behind House You are behind the white house. A path leads into the forest to the east. In one corner of the house there is a small window which is slightly ajar.

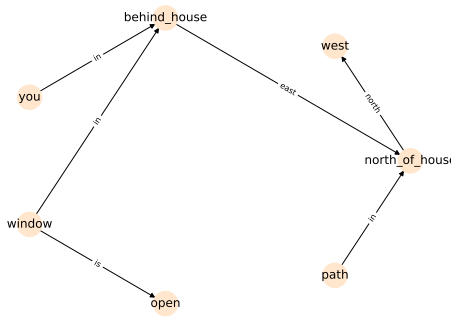
[inv] You are empty handed.

[obs] Behind House You are behind the white house. A path leads into the forest to the east. In one corner of the house there is a small window which is slightly ajar.

[atr] talkable, seen, lieable, enterable, nodwarf, indoors, visited, handed, lockable, surface, thing, water_room, unlock, lost, afflicted, is_treasure, converse, mentioned, male, npcworn, no_article, relevant, scored, queryable, town, pluggable, happy, is_followable, legible, multitude, burning, room, clothing, underneath, ward_area, little, intact, animate, bled_in, supporter, readable, openable, near, nonlocal, door, plugged, sittable, toolbit, vehicle, light, lens_searchable, open, familiar, is_scroll, aimable, takeable, static, unique, concealed, vowelstart, alcoholic, bodypart, general, is_spell, full, dry_land, pushable, known, proper, inside, clean, ambiguously_plural, container, edible, treasure, can_plug, weapon, is_arrow, insubstantial, pluralname, transparent, is_coin, air_room, scenery, on, is_spell_book, burnt, burnable, auto_searched, locked, switchable, absent, rockable, beenunlocked, progressing, severed, worn, windy, stone, random, neuter, legible, female, asleep, wiped

[graph] [(north_of_house, north, west), (behind_house, east, north_of_house), (you, in, behind_house), (door, is, animate), (door, in, west), (west, is, animate), (west, in, west), (you, in, behind_house), (mailbox, in, west), (mailbox, is, animate), (windows, in, north_of_house), (windows, is, animate), (north, is, animate), (north, in, north_of_house), (path, is, animate), (path, in, north_of_house), (trees, in, north_of_house), (trees, is, animate), (window, in, behind_house), (window, is, animate), (forest, in, behind_house), (forest, is, animate), (east, in, behind_house), (east, is, animate)]

[next act] open window

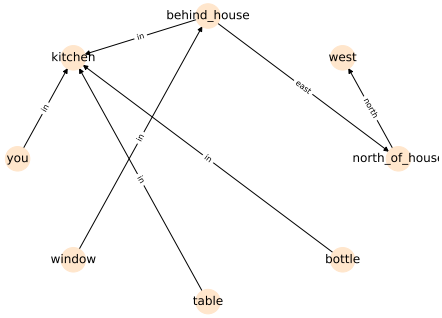


[loc] Behind House You are behind the white house. A path leads into the forest to the east. In one corner of the house there is a small window which is open. [inv] You are empty handed.

[obs] With great effort, you open the window far enough to allow entry.

[atr] talkable, seen, lieable, enterable, nodwarf, indoors, visited, handed, lockable, surface, thing, water_room, unlock, lost, afflicted, is_treasure, converse, mentioned, male, npcworn, no_article, relevant, scored, queryable, town, pluggable, happy, is_followable, legible, multitude, burning, room, clothing, underneath, ward_area, little, intact, animate, bled_in, supporter, readable, openable, near, nonlocal, door, plugged, sittable, toolbit, vehicle, light, lens_searchable, open, familiar, is_scroll, aimable, takeable, static, unique, concealed, vowelstart, alcoholic, bodypart, general, is_spell, full, dry_land, pushable, known, proper, inside, clean, ambiguously_plural, container, edible, treasure, can_plug, weapon, is_arrow, insubstantial, pluralname, transparent, is_coin, air_room, scenery, on, is_spell_book, burnt, burnable, auto_searched, locked, switchable, absent, rockable, beenunlocked, progressing, severed, worn, windy, stone, random, neuter, legible, female, asleep, wiped

[graph] [(north_of_house, north, west), (behind_house, east, north_of_house), (you, in, behind_house), (door, is, animate), (door, in, west), (west, is, animate), (west, in, west), (mailbox, in, west), (mailbox, is, animate), (windows, in, north_of_house), (windows, is, animate), (windows, is, open), (north, is, animate), (north, in, north_of_house), (path, is, animate), (path, in, north_of_house), (trees, in, north_of_house), (trees, is, animate), (window, in, behind_house), (window, is, animate), (forest, in, behind_house), (forest, is, animate), (east, in, behind_house), (east, is, animate)]



[loc] Kitchen You are in the kitchen of the white house. A table seems to have been used recently for the preparation of food. A passage leads to the west and a dark staircase can be seen leading upward. A dark chimney leads down and to the east is a small window which is open. On the table is an elongated brown sack, smelling of hot peppers. A bottle is sitting on the table. The glass bottle contains: A quantity of water

[inv] You are empty handed.

[obs] Kitchen You are in the kitchen of the white house. A table seems to have been used recently for the preparation of food. A passage leads to the west and a dark staircase can be seen leading upward. A dark chimney leads down and to the east is a small window which is open. On the table is an elongated brown sack, smelling of hot peppers. A bottle is sitting on the table. The glass bottle contains: A quantity of water

[atr] talkable, seen, lieable, enterable, nodwarf, indoors, visited, handed, lockable, surface, thing, water_room, unlock, lost, afflicted, is_treasure, converse, mentioned, male, npcworn, no_article, relevant, scored, queryable, town, pluggable, happy, is_followable, legible, multitude, burning, room, clothing, underneath, ward_area, little, intact, animate, bled_in, supporter, readable, openable, near, nonlocal, door, plugged, sittable, toolbit, vehicle, light, lens_searchable, open, familiar, is_scroll, aimable, takeable, static, unique, concealed, vowelstart, alcoholic, bodypart, general, is_spell, full, dry_land, pushable, known, proper, inside, clean, ambiguously_plural, container, edible, treasure, can_plug, weapon, is_arrow, insubstantial, pluralname, transparent, is_coin, air_room, scenery, on, is_spell_book, burnt, burnable, auto_searched, locked, switchable, absent, rockable, beenunlocked, progressing, severed, worn, windy, stone, random, neuter, legible, female, asleep, wiped

[graph] [(north_of_house, north, west), (behind_house, east, north_of_house), (behind_house, in, kitchen), (you, in, kitchen), (door, is, animate), (door, in, west), (west, is, animate), (west, in, west), (west, in, kitchen), (mailbox, in, west), (mailbox, is, animate), (windows, in, north_of_house), (windows, is, animate), (north, is, animate), (north, in, north_of_house), (path, is, animate), (path, in, north_of_house), (trees, in, north_of_house), (trees, is, animate), (window, in, behind_house), (window, is, animate), (forest, in, behind_house), (forest, is, animate), (east, in, behind_house), (east, is, animate), (table, in, kitchen), (table, is, animate)]]

[next act] go in

B.2.2 Architecture

The sequential action decoder consists two GRUs that are linked together as seen in earlier with KG-A2C. The first GRU decodes an action template and the second decodes objects that can be filled into the template. These objects are constrained by a *graph mask*, i.e. the decoder is only allowed to select entities that are already present in the knowledge graph.

The question answering network based on ALBERT (Lan *et al.* 2020) has the following hyperparameters, taken from the original paper and known to work well on the SQuAD 2.0 (Rajpurkar *et al.* 2018) dataset. No further hyperparameter tuning was conducted.

Parameters	**Value**
batch size	8
learning rate	3×10^{-5}
max seq len	512
doc stride	128
warmup steps	814
max steps	8144
gradient accumulation steps	24

B.2.3 MC!Q*BERT

The additional hyperparamters used for modular policy chaining are detailed below. *Patience batch factor* is the proportion of the batch that must have stagnated at a particular score for *patience* number of episodes of unchanging score before a bottleneck is detected. *Patience* within a range of $1000 - 6000$ in increments of 500 and *buffer size* within a range of $10 - 60$ in increments of 10 were the only additional parameters tuned for, on *Zork1*. The resulting best hyperparameter set was used on the rest of the games.

Parameters	Value
patience	3000
buffer size	40
batch size	16
patience batch factor	.75

B.2.4 GO!Q*BERT

Since the text games I are dealing with are mostly deterministic, with the exception of *Zork1* in later stages, I only focus on using Phase 1 of the Go-Explore algorithm to find an optimal policy. Go-Explore maintains an archive of cells—defined as a set of states that map to a single representation—to keep track of promising states. Ecoffet *et al.* 2021 simply encodes each cell by keeping track of the agent's position and Madotto *et al.* 2020 use the textual observations encoded by recurrent neural network as a cell representation. I improve on this implementation by training the Q*BERT network in parallel, using the snapshot of the knowledge graph in conjunction with the game state to further encode the current state and use this as a cell representation. At each step, Go-Explore chooses a cell to explore at random (weighted by score to prefer more advanced cells). Q*BERT will run for a number of steps in each cell, for all our experiments I use a cell step size of 32, starting with the knowledge graph state and the last seen state of the game from the cell. This will generate a trajectory for the agent while further training Q*BERT at each iteration, creating a new representation for the knowledge graph as well as a new game state for the cell. After expanding a cell, Go-Explore will continue to sample cells by weight to continue expanding its known states. At the same time, Q*BERT will benefit from the heuristics of selecting preferred cells and be trained on promising states more often.

B.2.5 Graph Evaluation Results

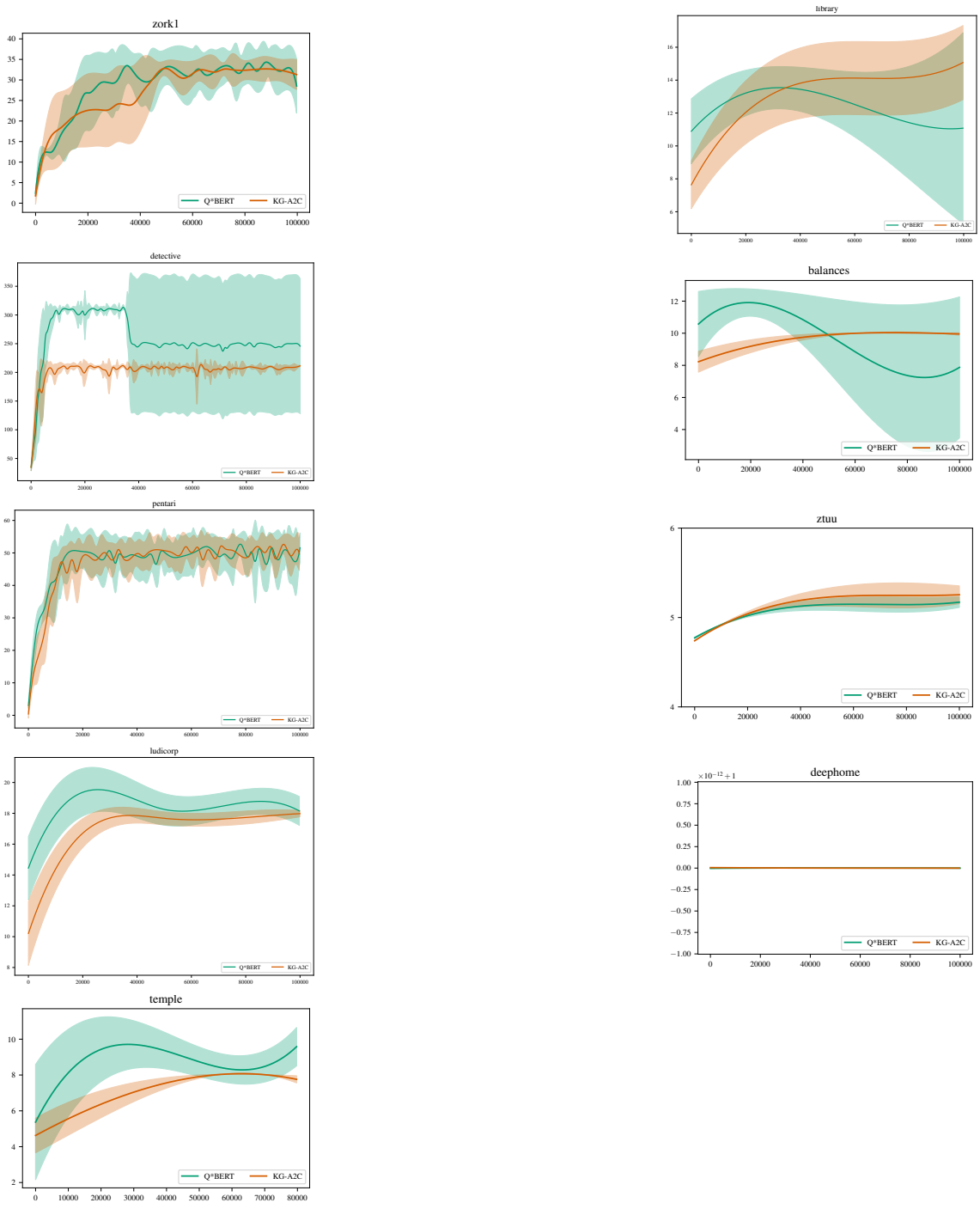


Figure B.2: Episode initial reward curves for KG-A2C and Q*BERT.

B.2.6 Intrinsic Motivation and Structured Exploration Results

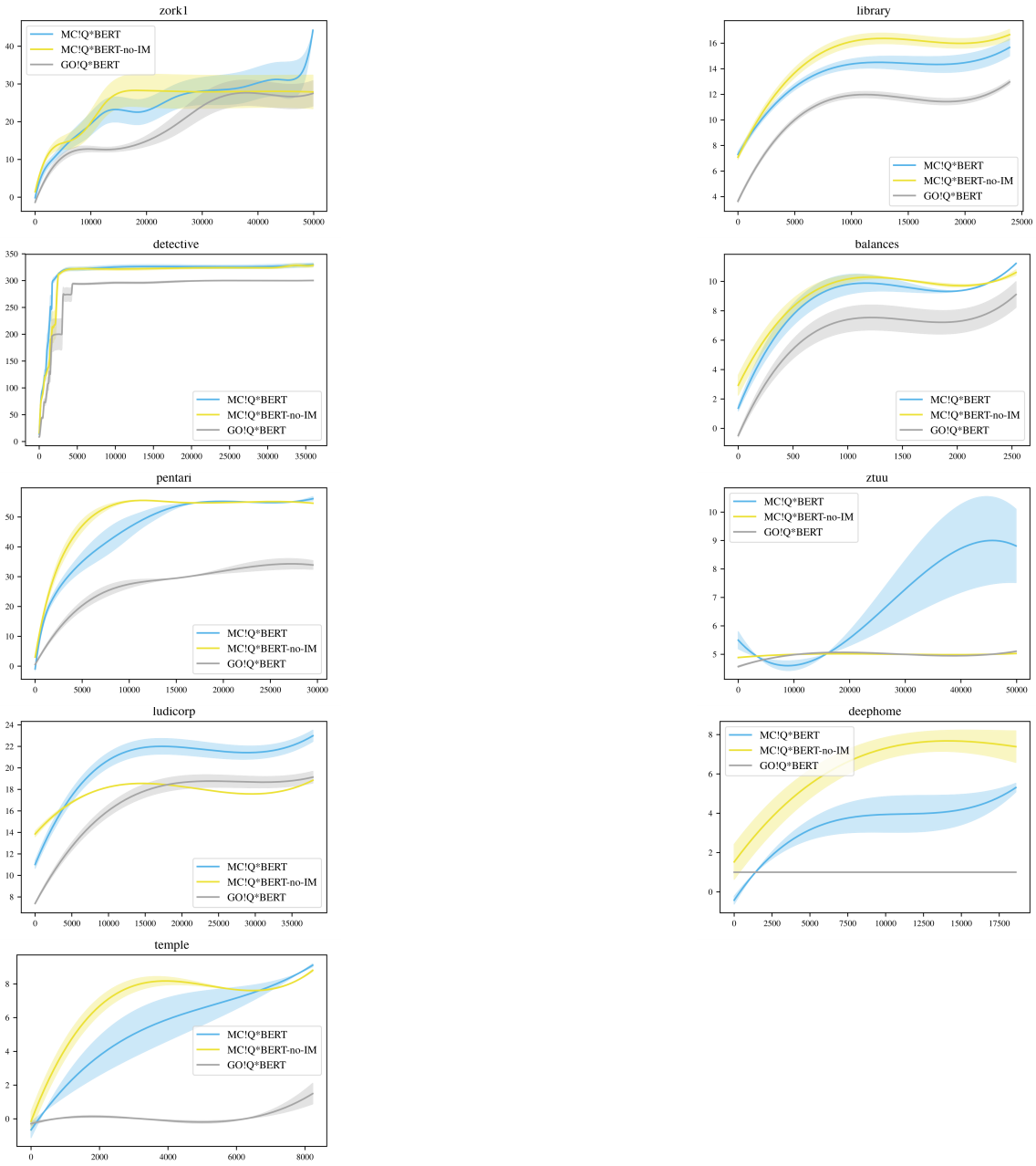


Figure B.3: Max initial reward curves for the exploration strategies.

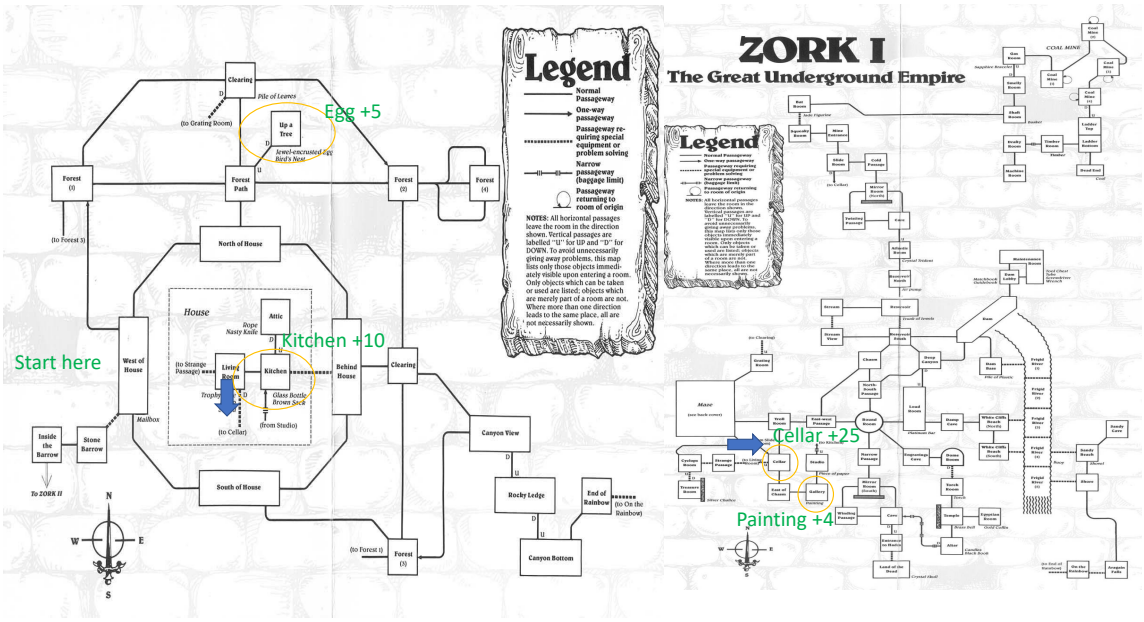


Figure B.4: A map of the world of *Zork1* with some initial rewards annotated. The blue arrow indicates a connection between the left and right maps, corresponding to the overworld and the dungeon.

B.3 Zork Agent Transcripts

Zork1 was identified by Hausknecht *et al.* 2020 to be one of the most difficult games in their suite and the subject of much prior work (Zahavy *et al.* 2018; Yin and May 2019b). *Zork1* is one of the earliest IF games and is a dungeon-crawler—a player must explore a vast labyrinth while fighting off enemies and complete puzzles in order to collect treasures. It features a relatively sparse reward for collecting a treasure or moving along the right path to one, and stochasticity in terms of random enemy movements.

To understand how humans progress in *Zork1*, a group of 10 human players—familiar with IF games—were asked to play *Zork1* for the first time (with no access to walkthroughs). Half of the players reached a game score of around 40 before dying to the first beatable NPC, a troll, mostly due to neglecting to collect a weapon to fight it with beforehand. Three of the remaining players died to hidden traps even before reaching this point, achieving scores between 5 and 15. The final two players made it significantly past

the troll gaining scores of around 70.

The following transcript of KG-A2C playing *Zork1*, shows top predicted probabilities for templates and objects.

```
Obs: Desc: 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 here. Inv: You are emptyhanded. Feedback:
     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 here.
Template probs: open OBJ: 0.339, east: 0.271, south: 0.215, west: 0.094, north: 0.031, go
    around OBJ: 0.013, blow out OBJ: 0.011, throw OBJ at OBJ: 0.01, northwest: 0.009
Object probs: window: 0.7, small: 0.299
Interactive objects: boarded, small, front, door, white, mailbox, house, all
Action: west
Score: 0
---
Obs: Desc: Forest This is a forest, with trees in all directions. To the east, there
    appears to be sunlight. Inv: You are emptyhanded. Feedback: Forest This is a forest,
    with trees in all directions. To the east, there appears to be sunlight.
Template probs: open OBJ: 0.49, south: 0.304, east: 0.116, west: 0.04, north: 0.016, go
    around OBJ: 0.011, throw OBJ at OBJ: 0.006, northwest: 0.006, blow out OBJ: 0.006
Object probs: window: 0.7, small: 0.3
Interactive objects: forest, all, trees
Action: east
Score: 0
---
Obs: Desc: Forest Path This is a path winding through a dimly lit forest. The path heads
    northsouth here. One particularly large tree with some low branches stands at the edge
     of the path. Inv: You are emptyhanded. Feedback: Forest Path This is a path winding
    through a dimly lit forest. The path heads northsouth here. One particularly large
    tree with some low branches stands at the edge of the path.
Template probs: up: 0.386, open OBJ: 0.358, east: 0.123, west: 0.06, north: 0.023, go
    around OBJ: 0.014, blow out OBJ: 0.01, northwest: 0.009, throw OBJ at OBJ: 0.008
Object probs: window: 0.701, small: 0.299
Interactive objects: tree, path, branches, forest, large, all
Action: up
Score: 0
---
Obs: Desc: Up a Tree You are about 10 feet above the ground nestled among some large
    branches. The nearest branch above you is above your reach. Beside you on the branch
    is a small birds nest. In the birds nest is a large egg encrusted with precious jewels
    , apparently scavenged by a childless songbird. The egg is covered with fine gold
```

inlay, and ornamented in lapis lazuli and motherofpearl. Unlike most eggs, this one is hinged and closed with a delicate looking clasp. The egg appears extremely fragile. Inv: You are emptyhanded. Feedback: Up a Tree You are about 10 feet above the ground nestled among some large branches. The nearest branch above you is above your reach. Beside you on the branch is a small birds nest. In the birds nest is a large egg encrusted with precious jewels, apparently scavenged by a childless songbird. The egg is covered with fine gold inlay, and ornamented in lapis lazuli and motherofpearl. Unlike most eggs, this one is hinged and closed with a delicate looking clasp. The egg appears extremely fragile.

Template probs: west: 0.43, open OBJ: 0.18, go around OBJ: 0.074, east: 0.064, north: 0.063, south: 0.052, take OBJ: 0.041, put down OBJ: 0.023, down: 0.014, close OBJ: 0.009

Object probs: small: 0.213, window: 0.184, house: 0.116, egg: 0.068, white: 0.067, path: 0.062, branch: 0.042, kitche: 0.024, nest: 0.018, ground: 0.009

Interactive objects: nest, large, egg, ground, branches, all, branch, tree

Action: open nest

Score: 0

Obs: Desc: Up a Tree You are about 10 feet above the ground nestled among some large branches. The nearest branch above you is above your reach. Beside you on the branch is a small birds nest. In the birds nest is a large egg encrusted with precious jewels, apparently scavenged by a childless songbird. The egg is covered with fine gold inlay, and ornamented in lapis lazuli and motherofpearl. Unlike most eggs, this one is hinged and closed with a delicate looking clasp. The egg appears extremely fragile. You hear in the distance the chirping of a song bird. Inv: You are emptyhanded. You hear in the distance the chirping of a song bird. Feedback: It is already open.

Template probs: west: 0.347, open OBJ: 0.182, go around OBJ: 0.085, east: 0.077, north: 0.073, south: 0.058, take OBJ: 0.045, put down OBJ: 0.026, down: 0.018, close OBJ: 0.013

Object probs: small: 0.213, window: 0.183, house: 0.117, egg: 0.068, white: 0.067, path: 0.062, branch: 0.042, kitche: 0.024, nest: 0.018, ground: 0.009

Interactive objects: nest, large, egg, ground, branches, all, branch, tree

Action: take egg

Score: 0

Obs: Desc: Up a Tree You are about 10 feet above the ground nestled among some large branches. The nearest branch above you is above your reach. Beside you on the branch is a small birds nest. Inv: You are carrying A jewelencrusted egg Feedback: Taken.

Template probs: down: 0.417, open OBJ: 0.181, go around OBJ: 0.074, east: 0.065, north: 0.063, south: 0.053, take OBJ: 0.042, put down OBJ: 0.025, down: 0.014, close OBJ: 0.01

Object probs: small: 0.213, window: 0.184, house: 0.116, egg: 0.068, white: 0.067, path: 0.062, branch: 0.042, kitche: 0.024, nest: 0.018, ground: 0.009

Interactive objects: nest, large, egg, ground, branches, all, branch, tree

Action: down

Score: 5

Obs: Desc: Forest Path This is a path winding through a dimly lit forest. The path heads northsouth here. One particularly large tree with some low branches stands at the edge of the path. Inv: You are emptyhanded. Feedback: Forest Path

Template probs: south: 0.386, open OBJ: 0.358, east: 0.123, west: 0.06, north: 0.023, go around OBJ: 0.014, blow out OBJ: 0.01, northwest: 0.009, throw OBJ at OBJ: 0.008

Object probs: window: 0.701, small: 0.299

Interactive objects: tree, path, branches, forest, large, all

Action: south

Score: 5

Obs: Desc: North of House You are facing the north side of a white house. There is no door here, and all the windows are boarded up. To the north a narrow path winds through the trees. Inv: You are emptyhanded. Feedback: North of House You are facing the north side of a white house. There is no door here, and all the windows are boarded up. To the north a narrow path winds through the trees.

Template probs: west: 0.502, open OBJ: 0.177, west: 0.147, north: 0.094, south: 0.035, throw OBJ at OBJ: 0.012, go around OBJ: 0.012, northwest: 0.01, blow out OBJ: 0.005

Object probs: window: 0.724, small: 0.243, trap: 0.014, kitche: 0.009, rug: 0.003

Interactive objects: path, windows, narrow, white, house, all

Action: west

Score: 5

Obs: Desc: 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 here. Inv: You are emptyhanded. Feedback:

Template probs: open OBJ: 0.536, south: 0.241, east: 0.121, west: 0.053, north: 0.015, go around OBJ: 0.011, throw OBJ at OBJ: 0.007, blow out OBJ: 0.006, northwest: 0.005

Object probs: window: 0.7, small: 0.3

Interactive objects: boarded, small, front, door, white, mailbox, house, all

Action: open white

Score: 5

Obs: Desc: 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 here. Inv: You are emptyhanded. Feedback: I cant see how to get in from here.

Template probs: open OBJ: 0.339, east: 0.271, south: 0.215, west: 0.094, north: 0.031, go

around OBJ: 0.013, blow out OBJ: 0.011, throw OBJ at OBJ: 0.01, northwest: 0.009
Object probs: window: 0.7, small: 0.299
Interactive objects: boarded, small, front, door, white, mailbox, house, all
Action: west
Score: 5

Obs: Desc: Forest This is a forest, with trees in all directions. To the east, there appears to be sunlight. Inv: You are emptyhanded. Feedback: Forest This is a forest, with trees in all directions. To the east, there appears to be sunlight.
Template probs: open OBJ: 0.49, south: 0.304, east: 0.116, west: 0.04, north: 0.016, go around OBJ: 0.011, throw OBJ at OBJ: 0.006, northwest: 0.006, blow out OBJ: 0.006
Object probs: window: 0.7, small: 0.3
Interactive objects: forest, all, trees
Action: east
Score: 5

Obs: Desc: Forest Path This is a path winding through a dimly lit forest. The path heads northsouth here. One particularly large tree with some low branches stands at the edge of the path. Inv: You are emptyhanded. Feedback: Forest Path This is a path winding through a dimly lit forest. The path heads northsouth here. One particularly large tree with some low branches stands at the edge of the path.
Template probs: south: 0.386, open OBJ: 0.358, east: 0.123, west: 0.06, north: 0.023, go around OBJ: 0.014, blow out OBJ: 0.01, northwest: 0.009, throw OBJ at OBJ: 0.008
Object probs: window: 0.701, small: 0.298
Interactive objects: tree, path, branches, forest, large, all
Action: open all
Score: 5

Obs: Desc: Forest Path This is a path winding through a dimly lit forest. The path heads northsouth here. One particularly large tree with some low branches stands at the edge of the path. Inv: You are emptyhanded. Feedback: Its not clear what youre referring to.
Template probs: open OBJ: 0.331, south: 0.259, east: 0.24, west: 0.087, north: 0.031, go around OBJ: 0.013, blow out OBJ: 0.011, northwest: 0.01, throw OBJ at OBJ: 0.009
Object probs: window: 0.701, small: 0.299
Interactive objects: tree, path, branches, forest, large, all
Action: east
Score: 5

Obs: Desc: Forest This is a dimly lit forest, with large trees all around. Inv: You are emptyhanded. You hear in the distance the chirping of a song bird. Feedback: Forest

This is a dimly lit forest, with large trees all around. You hear in the distance the chirping of a song bird.

Template probs: south: 0.38, open OBJ: 0.281, east: 0.174, west: 0.072, north: 0.034, go around OBJ: 0.015, blow out OBJ: 0.013, northwest: 0.012, throw OBJ at OBJ: 0.008

Object probs: window: 0.7, small: 0.299

Interactive objects: forest, large, all, trees

Action: south

Score: 5

Obs: Desc: Clearing You are in a small clearing in a well marked forest path that extends to the east and west. Inv: You are emptyhanded. Feedback: Clearing You are in a small clearing in a well marked forest path that extends to the east and west.

Template probs: open OBJ: 0.451, east: 0.364, west: 0.164, north: 0.008, throw OBJ at OBJ: 0.005

Object probs: window: 0.592, small: 0.405, kitche: 0.002

Interactive objects: forest, path, all

Action: east

Score: 5

Obs: Desc: Canyon View You are at the top of the Great Canyon on its west wall. From here there is a marvelous view of the canyon and parts of the Frigid River upstream. Across the canyon, the walls of the White Cliffs join the mighty ramparts of the Flathead Mountains to the east. Following the Canyon upstream to the north, Aragain Falls may be seen, complete with rainbow. The mighty Frigid River flows out from a great dark cavern. To the west and south can be seen an immense forest, stretching for miles around. A path leads northwest. It is possible to climb down into the canyon from here. Inv: You are emptyhanded. Feedback: Canyon View You are at the top of the Great Canyon on its west wall. From here there is a marvelous view of the canyon and parts of the Frigid River upstream. Across the canyon, the walls of the White Cliffs join the mighty ramparts of the Flathead Mountains to the east. Following the Canyon upstream to the north, Aragain Falls may be seen, complete with rainbow. The mighty Frigid River flows out from a great dark cavern. To the west and south can be seen an immense forest, stretching for miles around. A path leads northwest. It is possible to climb down into the canyon from here.

Template probs: west: 0.444, take OBJ: 0.204, open OBJ: 0.147, blow out OBJ: 0.118, turn on OBJ: 0.027, east: 0.026, down: 0.009, go around OBJ: 0.005, push OBJ: 0.004

Object probs: window: 0.693, small: 0.307

Interactive objects: path, walls, frigid, river, rainbow, wall, all

Action: west

Score: 5

Obs: Desc: Forest This is a dimly lit forest, with large trees all around. Inv: You are emptyhanded. Feedback: Forest This is a dimly lit forest, with large trees all around.
Template probs: south: 0.494, open OBJ: 0.241, east: 0.105, west: 0.053, north: 0.038, go around OBJ: 0.018, northwest: 0.015, blow out OBJ: 0.014, throw OBJ at OBJ: 0.007, down: 0.005
Object probs: window: 0.699, small: 0.301
Interactive objects: forest, large, all, trees
Action: south
Score: 5

Obs: Desc: Forest This is a dimly lit forest, with large trees all around. Inv: You are emptyhanded. Feedback: Stormtossed trees block your way.
Template probs: east: 0.722, north: 0.067, west: 0.061, open OBJ: 0.061, south: 0.048, blow out OBJ: 0.011, northwest: 0.01, go around OBJ: 0.007, throw OBJ at OBJ: 0.005
Object probs: window: 0.7, small: 0.3
Interactive objects: storm, forest, trees, large, all
Action: east
Score: 5

Obs: Desc: Forest This is a dimly lit forest, with large trees all around. Inv: You are emptyhanded. Feedback: The rank undergrowth prevents eastward movement.
Template probs: south: 0.434, open OBJ: 0.339, east: 0.076, west: 0.065, north: 0.024, go around OBJ: 0.02, northwest: 0.012, blow out OBJ: 0.011, throw OBJ at OBJ: 0.009
Object probs: window: 0.701, small: 0.299
Interactive objects: forest, large, all, trees
Action: jump
Score: 5

Obs: Desc: Forest This is a dimly lit forest, with large trees all around. Inv: You are emptyhanded. Feedback: Wheeeeeeeeee!!!!! You hear in the distance the chirping of a song bird.
Template probs: south: 0.438, open OBJ: 0.292, east: 0.098, west: 0.075, north: 0.03, go around OBJ: 0.019, northwest: 0.013, blow out OBJ: 0.012, throw OBJ at OBJ: 0.01
Object probs: window: 0.7, small: 0.299
Interactive objects: forest, large, all, trees
Action: south
Score: 5

Obs: Desc: Forest This is a dimly lit forest, with large trees all around. Inv: You are emptyhanded. Feedback: Stormtossed trees block your way.
Template probs: east: 0.694, west: 0.081, north: 0.07, open OBJ: 0.067, south: 0.043,

northwest: 0.012, blow out OBJ: 0.011, go around OBJ: 0.007, throw OBJ at OBJ: 0.006
Object probs: window: 0.7, small: 0.3
Interactive objects: storm, forest, trees, large, all
Action: north
Score: 5

Obs: Desc: Clearing You are in a small clearing in a well marked forest path that extends to the east and west. Inv: You are emptyhanded. Feedback: Clearing
Template probs: west: 0.437, east: 0.311, open OBJ: 0.171, north: 0.026, throw OBJ at OBJ: 0.015, south: 0.013, blow out OBJ: 0.011, go around OBJ: 0.006, northwest: 0.006
Object probs: window: 0.699, small: 0.301
Interactive objects: forest, path, all
Action: throw path at forest
Score: 5

Obs: Desc: Clearing You are in a small clearing in a well marked forest path that extends to the east and west. Inv: You are emptyhanded. Feedback: You dont have the way.
Template probs: open OBJ: 0.377, west: 0.341, east: 0.149, south: 0.057, north: 0.022, throw OBJ at OBJ: 0.018, blow out OBJ: 0.011, go around OBJ: 0.011, northwest: 0.008
Object probs: window: 0.7, small: 0.299
Interactive objects: forest, path, all
Action: west
Score: 5

Obs: Desc: Behind House You are behind the white house. A path leads into the forest to the east. In one corner of the house there is a small window which is slightly ajar. Inv: You are emptyhanded. Feedback: Behind House You are behind the white house. A path leads into the forest to the east. In one corner of the house there is a small window which is slightly ajar.
Template probs: open OBJ: 0.953, west: 0.036, south: 0.006
Object probs: small: 0.898, window: 0.102
Interactive objects: path, window, small, white, house, all
Action: open house
Score: 5

Obs: Desc: Behind House You are behind the white house. A path leads into the forest to the east. In one corner of the house there is a small window which is slightly ajar. Inv: You are emptyhanded. Feedback: The window is closed.
Template probs: open OBJ: 0.943, west: 0.047
Object probs: small: 0.88, window: 0.12
Interactive objects: path, window, small, white, house, all

Action: open path

Score: 5

Obs: Desc: Behind House You are behind the white house. A path leads into the forest to the east. In one corner of the house there is a small window which is slightly ajar. Inv: You are emptyhanded. Feedback: You must tell me how to do that to a way.

Template probs: open OBJ: 0.899, east: 0.048, west: 0.044, south: 0.004

Object probs: window: 0.58, small: 0.417, kitche: 0.002

Interactive objects: path, window, small, white, house, all

Action: open window

Score: 5

Obs: Desc: Behind House You are behind the white house. A path leads into the forest to the east. In one corner of the house there is a small window which is open. Inv: You are emptyhanded. Feedback: With great effort, you open the window far enough to allow entry.

Template probs: west: 0.997

Object probs: small: 0.951, window: 0.049

Interactive objects: path, window, small, white, house, all

Action: west

Score: 5

Obs: Desc: Kitchen You are in the kitchen of the white house. A table seems to have been used recently for the preparation of food. A passage leads to the west and a dark staircase can be seen leading upward. A dark chimney leads down and to the east is a small window which is open. On the table is an elongated brown sack, smelling of hot peppers. A bottle is sitting on the table. The glass bottle contains A quantity of water Inv: You are emptyhanded. Feedback: Kitchen You are in the kitchen of the white house. A table seems to have been used recently for the preparation of food. A passage leads to the west and a dark staircase can be seen leading upward. A dark chimney leads down and to the east is a small window which is open. On the table is an elongated brown sack, smelling of hot peppers. A bottle is sitting on the table. The glass bottle contains A quantity of water

Template probs: take OBJ: 0.47, west: 0.351, turn on OBJ: 0.061, blow out OBJ: 0.052, down : 0.023, open OBJ: 0.012, east: 0.008, close OBJ: 0.006, throw OBJ at OBJ: 0.006

Object probs: window: 0.683, small: 0.317

Interactive objects: kitchen, window, passage, staircase, glass, sack, water, small, table , chimney, bottle, quantity, brown, all

Action: west

Score: 15

Obs: Desc: Living Room You are in the living room. There is a doorway to the east, a wooden door with strange gothic lettering to the west, which appears to be nailed shut, a trophy case, and a large oriental rug in the center of the room. Above the trophy case hangs an elvish sword of great antiquity. A batterypowered brass lantern is on the trophy case. Inv: You are emptyhanded. Feedback: Living Room You are in the living room. There is a doorway to the east, a wooden door with strange gothic lettering to the west, which appears to be nailed shut, a trophy case, and a large oriental rug in the center of the room. Above the trophy case hangs an elvish sword of great antiquity. A batterypowered brass lantern is on the trophy case.

Template probs: take OBJ: 0.454, west: 0.241, turn on OBJ: 0.107, open OBJ: 0.053, down: 0.05, blow out OBJ: 0.034, close OBJ: 0.016, throw OBJ at OBJ: 0.011, east: 0.009, north: 0.005

Object probs: window: 0.685, small: 0.315

Interactive objects: elvish, strange, trophy, brass, wooden, rug, oriental, lettering, antiquity, sword, gothic, west, door, large, lantern, case, all

Action: take brass

Score: 15

Obs: Desc: Living Room You are in the living room. There is a doorway to the east, a wooden door with strange gothic lettering to the west, which appears to be nailed shut, a trophy case, and a large oriental rug in the center of the room. Above the trophy case hangs an elvish sword of great antiquity. Inv: You are carrying A brass lantern Feedback: Taken.

Template probs: open OBJ: 0.48, push OBJ: 0.138, west: 0.088, take OBJ: 0.067, turn on OBJ: 0.059, throw OBJ at OBJ: 0.031, put down OBJ: 0.023, put OBJ in OBJ: 0.018, put OBJ on OBJ: 0.016, down: 0.015

Object probs: window: 0.685, small: 0.315

Interactive objects: elvish, strange, trophy, brass, wooden, rug, oriental, lettering, antiquity, sword, gothic, west, door, large, lantern, case, all

Action: turn on brass

Score: 15

Obs: Desc: Living Room You are in the living room. There is a doorway to the east, a wooden door with strange gothic lettering to the west, which appears to be nailed shut, a trophy case, and a large oriental rug in the center of the room. Above the trophy case hangs an elvish sword of great antiquity. Inv: You are carrying A brass lantern (providing light) Feedback: The brass lantern is now on.

Template probs: open OBJ: 0.438, push OBJ: 0.141, west: 0.088, take OBJ: 0.081, turn on OBJ: 0.061, throw OBJ at OBJ: 0.036, put down OBJ: 0.023, put OBJ in OBJ: 0.019, down: 0.018, put OBJ on OBJ: 0.018

Object probs: window: 0.685, small: 0.315

Interactive objects: brass, strange, trophy, elvish, rug, oriental, light, lettering, antiquity, sword, gothic, west, door, large, wooden, lantern, case, all

Action: take trophy

Score: 15

Obs: Desc: Living Room You are in the living room. There is a doorway to the east, a wooden door with strange gothic lettering to the west, which appears to be nailed shut, a trophy case, and a large oriental rug in the center of the room. Above the trophy case hangs an elvish sword of great antiquity. Inv: You are carrying A brass lantern (providing light) Feedback: The trophy case is securely fastened to the wall.

Template probs: open OBJ: 0.437, push OBJ: 0.141, west: 0.087, take OBJ: 0.082, turn on OBJ: 0.061, throw OBJ at OBJ: 0.035, put down OBJ: 0.024, down: 0.019, put OBJ on OBJ: 0.018, put OBJ in OBJ: 0.018

Object probs: window: 0.684, small: 0.315

Interactive objects: strange, trophy, case, antiquity, lantern, lettering, sword, gothic, wooden, wall, elvish, brass, west, rug, oriental, door, large, light, all

Action: push case

Score: 15

Obs: Desc: Living Room You are in the living room. There is a doorway to the east, a wooden door with strange gothic lettering to the west, which appears to be nailed shut, a trophy case, and a large oriental rug in the center of the room. Above the trophy case hangs an elvish sword of great antiquity. Inv: You are carrying A brass lantern (providing light) Feedback: Pushing the trophy case has no effect.

Template probs: open OBJ: 0.407, push OBJ: 0.145, take OBJ: 0.09, west: 0.085, turn on OBJ: 0.062, throw OBJ at OBJ: 0.037, put down OBJ: 0.024, down: 0.021, put OBJ on OBJ: 0.019, put OBJ in OBJ: 0.019

Object probs: window: 0.686, small: 0.314

Interactive objects: elvish, strange, trophy, brass, wooden, rug, oriental, case, lettering, antiquity, sword, gothic, west, door, large, lantern, light, all

Action: push lanter

Score: 15

Obs: Desc: Living Room You are in the living room. There is a doorway to the east, a wooden door with strange gothic lettering to the west, which appears to be nailed shut, a trophy case, and a large oriental rug in the center of the room. Above the trophy case hangs an elvish sword of great antiquity. Inv: You are carrying A brass lantern (providing light) Feedback: Pushing the brass lantern isnt notably helpful.

Template probs: open OBJ: 0.405, push OBJ: 0.144, west: 0.093, take OBJ: 0.089, turn on OBJ: 0.062, throw OBJ at OBJ: 0.038, put down OBJ: 0.025, put OBJ on OBJ: 0.02, down: 0.019, put OBJ in OBJ: 0.019

Object probs: window: 0.685, small: 0.315

Interactive objects: brass, strange, trophy, elvish, rug, oriental, light, lettering, antiquity, sword, gothic, west, door, large, wooden, lantern, case, all

Action: open wooden

Score: 15

Obs: Desc: Living Room You are in the living room. There is a doorway to the east, a wooden door with strange gothic lettering to the west, which appears to be nailed shut, a trophy case, and a large oriental rug in the center of the room. Above the trophy case hangs an elvish sword of great antiquity. Inv: You are carrying A brass lantern (providing light) Feedback: The door cannot be opened.

Template probs: open OBJ: 0.438, push OBJ: 0.142, take OBJ: 0.081, west: 0.08, turn on OBJ: 0.062, throw OBJ at OBJ: 0.035, put down OBJ: 0.024, down: 0.02, put OBJ in OBJ: 0.018, put OBJ on OBJ: 0.018

Object probs: window: 0.685, small: 0.315

Interactive objects: elvish, strange, trophy, brass, wooden, rug, oriental, case, lettering, antiquity, sword, gothic, west, door, large, lantern, light, all

Action: open brass

Score: 15

Obs: Desc: Living Room You are in the living room. There is a doorway to the east, a wooden door with strange gothic lettering to the west, which appears to be nailed shut, a trophy case, and a large oriental rug in the center of the room. Above the trophy case hangs an elvish sword of great antiquity. Inv: You are carrying A brass lantern (providing light) Feedback: You must tell me how to do that to a brass lantern.

Template probs: open OBJ: 0.503, push OBJ: 0.129, west: 0.1, take OBJ: 0.053, turn on OBJ: 0.051, throw OBJ at OBJ: 0.041, put down OBJ: 0.029, put OBJ in OBJ: 0.021, put OBJ on OBJ: 0.02, down: 0.009

Object probs: window: 0.677, small: 0.323

Interactive objects: brass, strange, trophy, elvish, rug, oriental, light, lettering, antiquity, sword, gothic, west, door, large, wooden, lantern, case, all

Action: push rug

Score: 15

Obs: Desc: Living Room You are in the living room. There is a doorway to the east, a wooden door with strange gothic lettering to the west, which appears to be nailed shut, a trophy case, and a closed trap door at your feet. Above the trophy case hangs an elvish sword of great antiquity. Inv: You are carrying A brass lantern (providing light) Feedback: With a great effort, the rug is moved to one side of the room, revealing the dusty cover of a closed trap door.

Template probs: open OBJ: 0.582, push OBJ: 0.082, down: 0.074, take OBJ: 0.065, put down

OBJ: 0.025, turn on OBJ: 0.025, west: 0.023, close OBJ: 0.021, throw OBJ at OBJ: 0.019, put OBJ in OBJ: 0.016

Object probs: window: 0.674, small: 0.326

Interactive objects: dusty, strange, trophy, cover, case, antiquity, lantern, lettering, sword, gothic, wooden, elvish, brass, west, rug, trap, door, light, all

Action: west

Score: 15

Obs: Desc: Living Room You are in the living room. There is a doorway to the east, a wooden door with strange gothic lettering to the west, which appears to be nailed shut, a trophy case, and a closed trap door at your feet. Above the trophy case hangs an elvish sword of great antiquity. Inv: You are carrying A brass lantern (providing light) Feedback: The door is nailed shut.

Template probs: open OBJ: 0.679, push OBJ: 0.073, down: 0.057, take OBJ: 0.046, put down OBJ: 0.022, turn on OBJ: 0.019, west: 0.017, close OBJ: 0.015, throw OBJ at OBJ: 0.015, put OBJ in OBJ: 0.015

Object probs: window: 0.672, small: 0.328

Interactive objects: elvish, strange, trophy, brass, case, lettering, antiquity, sword, gothic, west, door, trap, wooden, lantern, light, all

Action: down

Score: 15

Obs: Desc: Living Room You are in the living room. There is a doorway to the east, a wooden door with strange gothic lettering to the west, which appears to be nailed shut, a trophy case, and a closed trap door at your feet. Above the trophy case hangs an elvish sword of great antiquity. Inv: You are carrying A brass lantern (providing light) Feedback: The trap door is closed.

Template probs: open OBJ: 0.63, down: 0.087, take OBJ: 0.06, put down OBJ: 0.045, push OBJ: 0.04, west: 0.027, turn on OBJ: 0.02, close OBJ: 0.018, throw OBJ at OBJ: 0.018, put OBJ in OBJ: 0.015

Object probs: window: 0.657, small: 0.342

Interactive objects: elvish, strange, trophy, brass, case, lettering, antiquity, sword, gothic, west, trap, door, wooden, lantern, light, all

Action: open sword

Score: 15

Obs: Desc: Living Room You are in the living room. There is a doorway to the east, a wooden door with strange gothic lettering to the west, which appears to be nailed shut, a trophy case, and a closed trap door at your feet. Above the trophy case hangs an elvish sword of great antiquity. Inv: You are carrying A brass lantern (providing light) Feedback: You must tell me how to do that to a sword.

Template probs: open OBJ: 0.625, down: 0.075, take OBJ: 0.053, push OBJ: 0.05, put down OBJ: 0.049, west: 0.028, turn on OBJ: 0.021, throw OBJ at OBJ: 0.02, close OBJ: 0.019, put OBJ in OBJ: 0.018

Object probs: window: 0.659, small: 0.34

Interactive objects: elvish, strange, trophy, brass, case, lettering, antiquity, sword, gothic, west, door, trap, wooden, lantern, light, all

Action: open case

Score: 15

Obs: Desc: Living Room You are in the living room. There is a doorway to the east, a wooden door with strange gothic lettering to the west, which appears to be nailed shut , a trophy case, and a closed trap door at your feet. Above the trophy case hangs an elvish sword of great antiquity. Inv: You are carrying A brass lantern (providing light) Feedback: Opened.

Template probs: open OBJ: 0.61, down: 0.081, take OBJ: 0.056, push OBJ: 0.053, put down OBJ: 0.049, west: 0.029, turn on OBJ: 0.023, throw OBJ at OBJ: 0.02, put OBJ in OBJ: 0.019, close OBJ: 0.019

Object probs: window: 0.659, small: 0.341

Interactive objects: elvish, strange, trophy, brass, case, lettering, antiquity, sword, gothic, west, door, trap, wooden, lantern, light, all

Action: open door

Score: 15

Obs: Desc: Living Room You are in the living room. There is a doorway to the east, a wooden door with strange gothic lettering to the west, which appears to be nailed shut , a trophy case, and a closed trap door at your feet. Above the trophy case hangs an elvish sword of great antiquity. Inv: You are carrying A brass lantern (providing light) Feedback: Which door do you mean, the wooden door or the trap door?

Template probs: open OBJ: 0.552, down: 0.085, take OBJ: 0.062, put down OBJ: 0.06, push OBJ: 0.055, west: 0.04, turn on OBJ: 0.026, throw OBJ at OBJ: 0.026, put OBJ in OBJ: 0.024, close OBJ: 0.021

Object probs: window: 0.66, small: 0.33

Interactive objects: elvish, strange, trophy, brass, case, lettering, antiquity, sword, gothic, west, trap, door, wooden, lantern, light, all

Action: open all

Score: 15

Obs: Desc: Living Room You are in the living room. There is a doorway to the east, a wooden door with strange gothic lettering to the west, which appears to be nailed shut , a trophy case, and a closed trap door at your feet. Above the trophy case hangs an elvish sword of great antiquity. Inv: You are carrying A brass lantern (providing

light) Feedback: You cant use multiple direct objects with "open".

Template probs: open OBJ: 0.529, down: 0.086, put down OBJ: 0.065, take OBJ: 0.063, push OBJ: 0.059, west: 0.044, throw OBJ at OBJ: 0.028, turn on OBJ: 0.028, put OBJ in OBJ: 0.026, close OBJ: 0.021

Object probs: window: 0.661, small: 0.339

Interactive objects: elvish, strange, trophy, brass, case, lettering, antiquity, sword, gothic, west, door, trap, wooden, lantern, light, all

Action: down

Score: 15

Obs: Desc: Living Room You are in the living room. There is a doorway to the east, a wooden door with strange gothic lettering to the west, which appears to be nailed shut , a trophy case, and a closed trap door at your feet. Above the trophy case hangs an elvish sword of great antiquity. Inv: You are carrying A brass lantern (providing light) Feedback: The trap door is closed.

Template probs: open OBJ: 0.569, down: 0.089, take OBJ: 0.063, put down OBJ: 0.059, push OBJ: 0.048, west: 0.036, turn on OBJ: 0.024, throw OBJ at OBJ: 0.024, put OBJ in OBJ: 0.021, close OBJ: 0.021

Object probs: window: 0.658, small: 0.341

Interactive objects: elvish, strange, trophy, brass, case, lettering, antiquity, sword, gothic, west, trap, door, wooden, lantern, light, all

Action: shake gothic

Score: 15

Obs: Desc: Living Room You are in the living room. There is a doorway to the east, a wooden door with strange gothic lettering to the west, which appears to be nailed shut , a trophy case, and a closed trap door at your feet. Above the trophy case hangs an elvish sword of great antiquity. Inv: You are carrying A brass lantern (providing light) Feedback: You dont have the wooden door.

Template probs: open OBJ: 0.62, push OBJ: 0.085, take OBJ: 0.055, down: 0.053, west: 0.031, put down OBJ: 0.026, turn on OBJ: 0.025, throw OBJ at OBJ: 0.02, put OBJ in OBJ : 0.019, close OBJ: 0.014

Object probs: window: 0.668, small: 0.332

Interactive objects: elvish, strange, trophy, brass, case, lettering, antiquity, sword, gothic, west, door, trap, wooden, lantern, light, all

Action: open west

Score: 15

Obs: Desc: Living Room You are in the living room. There is a doorway to the east, a wooden door with strange gothic lettering to the west, which appears to be nailed shut , a trophy case, and a closed trap door at your feet. Above the trophy case hangs an

elvish sword of great antiquity. Inv: You are carrying A brass lantern (providing light) Feedback: The door cannot be opened.

Template probs: open OBJ: 0.648, push OBJ: 0.089, down: 0.053, take OBJ: 0.042, west: 0.025, turn on OBJ: 0.025, put down OBJ: 0.024, throw OBJ at OBJ: 0.021, put OBJ in OBJ: 0.02, close OBJ: 0.013

Object probs: window: 0.665, small: 0.335

Interactive objects: elvish, strange, trophy, brass, case, lettering, antiquity, sword, gothic, west, door, trap, wooden, lantern, light, all

Action: open wooden

Score: 15

Obs: Desc: Living Room You are in the living room. There is a doorway to the east, a wooden door with strange gothic lettering to the west, which appears to be nailed shut , a trophy case, and a closed trap door at your feet. Above the trophy case hangs an elvish sword of great antiquity. Inv: You are carrying A brass lantern (providing light) Feedback: The door cannot be opened.

Template probs: open OBJ: 0.637, push OBJ: 0.081, down: 0.058, take OBJ: 0.05, put down OBJ: 0.024, west: 0.024, turn on OBJ: 0.024, close OBJ: 0.019, throw OBJ at OBJ: 0.017, put OBJ in OBJ: 0.016

Object probs: window: 0.671, small: 0.329

Interactive objects: elvish, strange, trophy, brass, case, lettering, antiquity, sword, gothic, west, door, trap, wooden, lantern, light, all

Action: push sword

Score: 15

Obs: Desc: Living Room You are in the living room. There is a doorway to the east, a wooden door with strange gothic lettering to the west, which appears to be nailed shut , a trophy case, and a closed trap door at your feet. Above the trophy case hangs an elvish sword of great antiquity. Inv: You are carrying A brass lantern (providing light) Feedback: Pushing the sword doesnt seem to work.

Template probs: open OBJ: 0.591, push OBJ: 0.085, down: 0.073, take OBJ: 0.061, turn on OBJ: 0.025, put down OBJ: 0.024, west: 0.024, close OBJ: 0.02, throw OBJ at OBJ: 0.019, put OBJ in OBJ: 0.017

Object probs: window: 0.674, small: 0.326

Interactive objects: elvish, strange, trophy, brass, case, lettering, antiquity, sword, gothic, west, door, trap, wooden, lantern, light, all

Action: open strang

Score: 15

Obs: Desc: Living Room You are in the living room. There is a doorway to the east, a wooden door with strange gothic lettering to the west, which appears to be nailed shut

, a trophy case, and a closed trap door at your feet. Above the trophy case hangs an elvish sword of great antiquity. Inv: You are carrying A brass lantern (providing light) Feedback: The door cannot be opened.

Template probs: open OBJ: 0.614, push OBJ: 0.085, down: 0.065, take OBJ: 0.059, west: 0.024, turn on OBJ: 0.024, put down OBJ: 0.021, close OBJ: 0.017, throw OBJ at OBJ: 0.017, put OBJ in OBJ: 0.016

Object probs: window: 0.676, small: 0.323

Interactive objects: elvish, strange, trophy, brass, case, lettering, antiquity, sword, gothic, west, door, trap, wooden, lantern, light, all

Action: down

Score: 15

Obs: Desc: Living Room You are in the living room. There is a doorway to the east, a wooden door with strange gothic lettering to the west, which appears to be nailed shut , a trophy case, and a closed trap door at your feet. Above the trophy case hangs an elvish sword of great antiquity. Inv: You are carrying A brass lantern (providing light) Feedback: The trap door is closed.

Template probs: open OBJ: 0.665, down: 0.068, push OBJ: 0.052, take OBJ: 0.044, put down OBJ: 0.036, west: 0.026, turn on OBJ: 0.02, throw OBJ at OBJ: 0.018, put OBJ in OBJ: 0.017, close OBJ: 0.015

Object probs: window: 0.661, small: 0.339

Interactive objects: elvish, strange, trophy, brass, case, lettering, antiquity, sword, gothic, west, trap, door, wooden, lantern, light, all

Action: open trap

Score: 15

Obs: Desc: Living Room You are in the living room. There is a doorway to the east, a wooden door with strange gothic lettering to the west, which appears to be nailed shut , a trophy case, and a rug lying beside an open trap door. Above the trophy case hangs an elvish sword of great antiquity. Inv: You are carrying A brass lantern (providing light) Feedback: The door reluctantly opens to reveal a rickety staircase descending into darkness.

Template probs: down: 0.876, open OBJ: 0.046, take OBJ: 0.018, east: 0.013, close OBJ: 0.009, put down OBJ: 0.007, north: 0.006

Object probs: window: 0.604, small: 0.387, kitche: 0.006, trap: 0.003

Interactive objects: elvish, strange, trophy, brass, rug, staircase, case, lettering, antiquity, sword, gothic, west, door, trap, wooden, lantern, light, all

Action: down

Score: 15

Obs: Desc: It is pitch black. You are likely to be eaten by a grue. Inv: You are carrying

A brass lantern Feedback: You have moved into a dark place. It is pitch black. You are likely to be eaten by a grue.

Template probs: take OBJ: 0.141, turn on OBJ: 0.14, open OBJ: 0.122, down: 0.117, close OBJ: 0.073, put down OBJ: 0.069, west: 0.065, push OBJ: 0.056, throw OBJ at OBJ: 0.054, south: 0.02

Object probs: lanter: 0.076, letter: 0.071, window: 0.061, door: 0.059, small: 0.052, gothic: 0.049, sword: 0.049, rug: 0.037, trophy: 0.035, west: 0.031

Interactive objects: brass, dark, lantern, black, all, grue

Action: down

Score: 40

Obs: Desc: It is pitch black. You are likely to be eaten by a grue. Inv: You are carrying A brass lantern Feedback: The chasm probably leads straight to the infernal regions.

Template probs: turn on OBJ: 0.167, take OBJ: 0.132, open OBJ: 0.125, down: 0.124, close OBJ: 0.076, push OBJ: 0.067, put down OBJ: 0.066, throw OBJ at OBJ: 0.056, west: 0.05, south: 0.024

Object probs: lanter: 0.077, letter: 0.071, window: 0.06, door: 0.059, small: 0.051, gothic: 0.05, sword: 0.049, rug: 0.037, trophy: 0.035, west: 0.031

Interactive objects: brass, chasm, lantern, black, all, grue

Action: turn on lanter

Score: 40

Obs: Desc: Cellar You are in a dark and damp cellar with a narrow passageway leading north, and a crawlway to the south. On the west is the bottom of a steep metal ramp which is unclimbable. There is a suspiciouslooking individual, holding a large bag, leaning against one wall. He is armed with a deadly stiletto. Inv: You are carrying A brass lantern (providing light) Feedback: You try to ascend the ramp, but it is impossible, and you slide back down.

Template probs: down: 0.246, open OBJ: 0.209, take OBJ: 0.122, close OBJ: 0.1, push OBJ: 0.082, turn on OBJ: 0.068, put down OBJ: 0.033, south: 0.029, west: 0.019, throw OBJ at OBJ: 0.016

Object probs: west: 0.08, small: 0.08, window: 0.073, door: 0.067, rug: 0.057, letter: 0.047, wooden: 0.045, gothic: 0.041, strang: 0.035, lanter: 0.034

Interactive objects: light, ramp, large, brass, stiletto, crawlway, dark, narrow, lantern, bag, suspicious, metal, all, wall, passageway

Action: take dark

Score: 40

Obs: Desc: Cellar You are in a dark and damp cellar with a narrow passageway leading north, and a crawlway to the south. On the west is the bottom of a steep metal ramp which is unclimbable. There is a suspiciouslooking individual, holding a large bag, leaning

against one wall. He is armed with a deadly stiletto. Inv: You are carrying A brass lantern (providing light) Feedback: You cant be serious.

Template probs: down: 0.289, open OBJ: 0.201, take OBJ: 0.103, close OBJ: 0.091, turn on OBJ: 0.066, push OBJ: 0.065, put down OBJ: 0.043, throw OBJ at OBJ: 0.022, south: 0.021, west: 0.018

Object probs: west: 0.081, small: 0.076, window: 0.069, door: 0.068, rug: 0.057, letter: 0.048, wooden: 0.045, gothic: 0.042, strang: 0.036, lanter: 0.035

Interactive objects: light, ramp, large, brass, stiletto, crawlway, dark, narrow, lantern, bag, suspicious, metal, all, wall, passageway

Action: put out passag

Score: 40

Obs: Desc: Cellar You are in a dark and damp cellar with a narrow passageway leading north, and a crawlway to the south. On the west is the bottom of a steep metal ramp which is unclimbable. There is a suspiciouslooking individual, holding a large bag, leaning against one wall. He is armed with a deadly stiletto. The thief, finding nothing of value, left disgusted. Inv: You are carrying A brass lantern (providing light) The thief, finding nothing of value, left disgusted. Feedback: Pushing the chute isnt notably helpful. Someone carrying a large bag is casually leaning against one of the walls here. He does not speak, but it is clear from his aspect that the bag will be taken only over his dead body.

Template probs: open OBJ: 0.211, take OBJ: 0.205, push OBJ: 0.188, south: 0.132, west: 0.105, close OBJ: 0.057, turn on OBJ: 0.026, go around OBJ: 0.02, east: 0.017, down: 0.016

Object probs: small: 0.112, window: 0.103, west: 0.075, door: 0.063, rug: 0.054, wooden: 0.044, letter: 0.04, gothic: 0.037, strang: 0.032, kitche: 0.031

Interactive objects: light, ramp, large, brass, stiletto, crawlway, walls, dark, narrow, thief, lantern, chute, bag, suspicious, metal, all, wall, passageway

Action: west

Score: 40

Obs: Desc: Cellar You are in a dark and damp cellar with a narrow passageway leading north, and a crawlway to the south. On the west is the bottom of a steep metal ramp which is unclimbable. There is a suspiciouslooking individual, holding a large bag, leaning against one wall. He is armed with a deadly stiletto. A quick thrust pinks your left arm, and blood starts to trickle down. Inv: You are carrying A brass lantern (providing light) The thief, finding nothing of value, left disgusted. Feedback: You dont have the way.

Template probs: take OBJ: 0.246, open OBJ: 0.211, push OBJ: 0.19, south: 0.091, west: 0.082, close OBJ: 0.059, turn on OBJ: 0.041, down: 0.026, go around OBJ: 0.017, east: 0.012

Object probs: small: 0.097, window: 0.087, west: 0.078, door: 0.065, rug: 0.056, wooden: 0.045, letter: 0.043, gothic: 0.038, strang: 0.034, lanter: 0.032

Interactive objects: light, ramp, large, brass, stiletto, crawlway, dark, narrow, lantern, bag, suspicious, metal, all, wall, passageway

Action: south

Score: 40

Obs: Desc: East of Chasm You are on the east edge of a chasm, the bottom of which cannot be seen. A narrow passage goes north, and the path you are on continues to the east. Inv: You are carrying A brass lantern (providing light) Feedback: The brass lantern is now on. East of Chasm You are on the east edge of a chasm, the bottom of which cannot be seen. A narrow passage goes north, and the path you are on continues to the east.

Template probs: put down OBJ: 0.132, take OBJ: 0.116, east: 0.1, open OBJ: 0.093, close OBJ: 0.09, turn on OBJ: 0.07, throw OBJ at OBJ: 0.061, west: 0.059, push OBJ: 0.057, down: 0.042

Object probs: window: 0.074, small: 0.057, lanter: 0.046, sword: 0.041, gothic: 0.034, wooden: 0.031, letter: 0.03, bottle: 0.03, door: 0.028, large: 0.028

Interactive objects: light, brass, chasm, path, narrow, lantern, passage, all

Action: east

Score: 40

Obs: Desc: Gallery This is an art gallery. Most of the paintings have been stolen by vandals with exceptional taste. The vandals left through either the north or west exits. Fortunately, there is still one chance for you to be a vandal, for on the far wall is a painting of unparalleled beauty. Inv: You are emptyhanded. Feedback: Gallery This is an art gallery. Most of the paintings have been stolen by vandals with exceptional taste. The vandals left through either the north or west exits. Fortunately, there is still one chance for you to be a vandal, for on the far wall is a painting of unparalleled beauty.

Template probs: open OBJ: 0.42, west: 0.377, take OBJ: 0.063, blow out OBJ: 0.039, east: 0.026, push OBJ: 0.02, down: 0.016, go around OBJ: 0.01, north: 0.008, turn on OBJ: 0.006

Object probs: window: 0.701, small: 0.299

Interactive objects: art, painting, wall, all, paintings

Action: open painti

Score: 40

Obs: Desc: Gallery This is an art gallery. Most of the paintings have been stolen by vandals with exceptional taste. The vandals left through either the north or west exits. Fortunately, there is still one chance for you to be a vandal, for on the far wall is a painting of unparalleled beauty. Inv: You are emptyhanded. Feedback: You

```
    must tell me how to do that to a painting.
Template probs: open OBJ: 0.41, west: 0.33, take OBJ: 0.106, push OBJ: 0.033, blow out OBJ
    : 0.032, down: 0.023, turn on OBJ: 0.015, east: 0.014, go around OBJ: 0.01, south:
    0.006
Object probs: window: 0.701, small: 0.299
Interactive objects: art, painting, wall, all, paintings
Action: take painti
Score: 40
---
Obs: Desc: Gallery This is an art gallery. Most of the paintings have been stolen by
    vandals with exceptional taste. The vandals left through either the north or west
    exits. Inv: You are carrying A painting Feedback: Taken.
Template probs: open OBJ: 0.492, push OBJ: 0.296, take OBJ: 0.036, west: 0.033, turn on
    OBJ: 0.032, throw OBJ at OBJ: 0.021, down: 0.015, close OBJ: 0.013, put down OBJ:
    0.011, put OBJ in OBJ: 0.01
Object probs: window: 0.698, small: 0.302
Interactive objects: art, painting, all, paintings
Action: open art
Score: 44
---
```

B.4 LIGHT-Quests

Mechanical Turk Data Collection

Crowdworkers are required to first pass an on-boarding test before they are allowed to perform the actual task. Figures B.5, B.6, B.7, B.8, and B.9 describe first the instructions given to the crowdworkers and then 4 phases of the on-boarding test. I paid workers $2.75 per task. This amount was determined by first running the task ourselves to estimate a completion time of 10-12 minutes per task, and then running pilot tasks that confirmed the average task duration for workers was close to 10 minutes.

Figure B.10 shows the example of the actual task given to the crowdworkers and Figure B.11 shows the user interface for the first phase of the LIGHT-Quests data collection task.

Onboarding Test:

Page (0/4) - Attempts (0/7)

In this task you will be given a character, like the following:

Character Name: Farmer

Character Description: I am a very hard worker. I am up with the sun 7 days a week. I learned all I know from my father. I will leave my farm to my son.

Setting: You are in a field beside the farm house. The sun burns brightly in the sky, and there are crops all around. There is a shovel and a sack of unplanted seeds here.

Goal Action: get shovel

In order to complete the task properly, you'll need to read the character and description and pretend that you are playing as that character. You are also given an action that your character is about to take. Your core tasks will be to come up with a reason for why your character is taking the action given. It should be in-character, and make sense along with the rest of the details you have been given.

Next Page

Figure B.5: On-boarding test instructions.

Page (1/4) - Attempts (0/7)

The first three questions you will be asked about your task will relate to coming up with a motivation for why your character is taking that action. All of these motivations should be written in **first person** . The first should be something that can be resolved within a few minutes, perhaps by the specific action. A few good examples:

- **get apple:** I'm getting pretty hungry and would like a snack.
- **go outside:** I'd like to get a breath of fresh air.
- **wield sword:** I must prepare for an impending battle.

The reasoning that you come up with should be in-line with your given character, so a *princess* likely wouldn't provide *I'm hoping to sell some candy at the store* as a reason to *get chocolate* .

Imagine you're answering for the following character:

Character Name: Farmer

Character Description: I am a very hard worker. I am up with the sun 7 days a week. I learned all I know from my father. I will leave my farm to my son.

Setting: You are in a field beside the farm house. The sun burns brightly in the sky, and there are crops all around. There is a shovel and a sack of unplanted seeds here.

Goal Action: get shovel

1. Provide a short-term goal (completable within 10 minutes) from the first person perspective of your given character that the action "get shovel" is a step in completing. Ensure the goal makes sense given the rest of the character description and the current setting:

Which of the following answers to this question 1 are acceptable?

- [x] I'm planning to dig a hole.
- [ ] The farmer's shovel needs to be washed.
- [x] My neighbor would like to borrow my shovel.
- [x] I need a shovel for work today.
- [ ] I hope to open a business for selling shovels someday.
- [ ] I want to cut down some trees in this forest.

Submit answers and move to next page

Figure B.6: Phase 1 of the on-boarding test.

Page (2/4) - Attempts (1/7)

The second question should be something that can be resolved within a few hours to a day, and your response to question 1 should be a small part of this motivating goal. A few good examples:

- **I'd like to head to the market:** I feel like today's a good day to shop for new clothes.
- **I'm getting pretty hungry and would like a snack:** I've always wanted to hike this nearby trail.
- **I must prepare for an impending battle:** Every night the raiders arrive and I have to be ready.

Imagine you're answering for the following character:

Character Name: Farmer

Character Description: I am a very hard worker. I am up with the sun 7 days a week. I learned all I know from my father. I will leave my farm to my son.

Setting: You are in a field beside the farm house. The sun burns brightly in the sky, and there are crops all around. There is a shovel and a sack of unplanted seeds here.

Goal Action: get shovel

1. Provide a short-term goal (completable within 10 minutes) from the first person perspective of your given character that the action "get shovel" is a step in completing. Ensure the goal makes sense given the rest of the character description and the current setting:

I need a shovel to dig a hole

2. Provide a mid-term goal/motivation (completable in the range from hours to a few days) from first person perspective where accomplishing your answer to question 1 is a step towards completing this medium goal:

Which of the following answers to question 2 are acceptable?

- [ ] I want to dig a single hole.
- [x] I have to plant a new batch of potatos.
- [x] I'd like to hide family heirlooms from the king's corrupt tax collectors.
- [ ] A hole is required.
- [ ] I'm trying to pretend like I'm busy.

Submit answers and move to next page

Figure B.7: Phase 2 of the on-boarding test.

Page (3/4) - Attempts (1/7)

The first three questions you will be asked about your task will relate to coming up with a motivation for why your character is taking that action. All of these motivations should be written in **first person** . The first should be something that can be resolved within a few minutes, perhaps by the specific action. A few good examples:

- **I feel like today's a good day to shop for new clothes:** I'm trying to find ways to catch the local bartender's eye.
- **I've always wanted to hike this nearby trail:** I'm turning over a new leaf to get more in-touch with nature.
- **Every night the raiders arrive and I have to be ready:** I must protect the tomb's artifacts until the knights reinforcements arrive next month.

Imagine you're answering for the following character:

Character Name: Farmer

Character Description: I am a very hard worker. I am up with the sun 7 days a week. I learned all I know from my father. I will leave my farm to my son.

Setting: You are in a field beside the farm house. The sun burns brightly in the sky, and there are crops all around. There is a shovel and a sack of unplanted seeds here.

Goal Action: get shovel

1. Provide a short-term goal (completable within 10 minutes) from the first person perspective of your given character that the action "get shovel" is a step in completing. Ensure the goal makes sense given the rest of the character description and the current setting:

I need a shovel to dig a hole

2. Provide a mid-term goal/motivation (completable in the range from hours to a few days) from first person perspective where accomplishing your answer to question 1 is a step towards completing this medium goal:

I have to plant some potato seeds today

3. Provide a long-term goal/motivation (completable in the range from weeks to a few months) from the first person perspective where accomplishing your answer to question 2 is a part of completing this larger goal:

Which of the following answers to question 3 are acceptable?

- ☐ I think potatoes are fun.
- ☑ I need to grow some staple foods to feed my family.
- ☑ The potato festival is next month and I need to prepare.
- ☑ My son hasn't yet learned how to grow potatoes and I need to teach the whole process.
- ☐ I need to set up some new irrigation routes for the farm.

Submit answers and move to next page

Figure B.8: Phase 3 of the on-boarding test.

Page (4/4) - Attempts (1/7)

The last question asks you to fill out some possible actions that your character will take before and then after your given action. You should select an action type and then fill in only what the template requires. This will usually be just a single object, character, or location. You should **not** additionally provide context along with this action.

Imagine you're answering for the following character:

Character Name: Farmer

Character Description: I am a very hard worker. I am up with the sun 7 days a week. I learned all I know from my father. I will leave my farm to my son.

Setting: You are in a field beside the farm house. The sun burns brightly in the sky, and there are crops all around. There is a shovel and a sack of unplanted seeds here.

Goal Action: get shovel

And you've given the following motivations in questions 1-3:

I need a shovel so I can dig a hole. I have to plant some potato seeds today. I'm preparing for the potato festival and will need a lot for everyone in town.

Which of the following actions are acceptable to happen in the next 5 minutes from "get shovel"?

- ☐ Wield — shovel so I can start digging
- ☑ Wear — working boots
- ☑ Get — bag of seeds .
- ☐ Get — seeds and other items .
- ☐ Get — I get the seeds .
- ☑ Use — shovel with ground

Submit answers and move to next page

Figure B.9: Phase 4 of the on-boarding test.

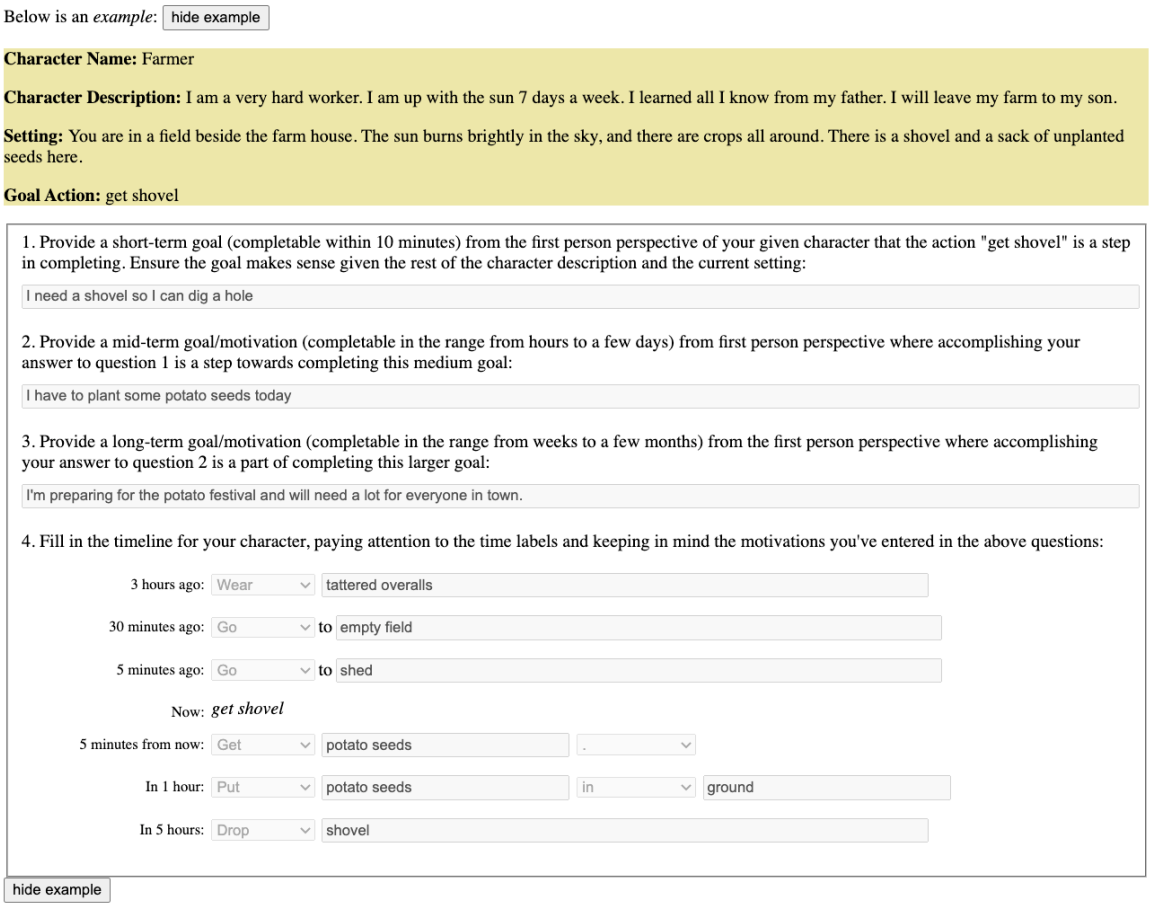


Figure B.10: Example for the first phase of the LIGHT-Quests data collection task given to the crowdworkers.

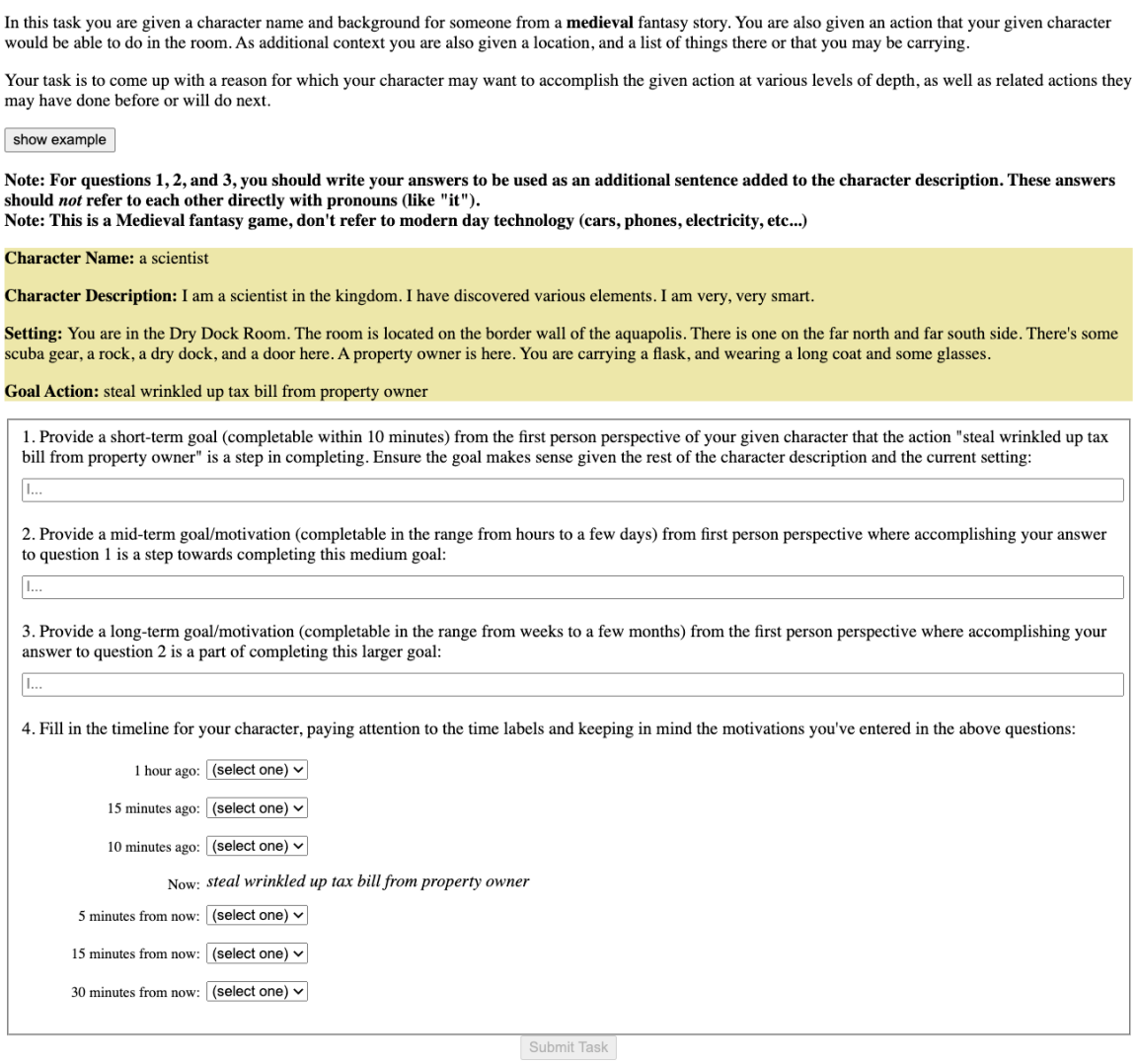


Figure B.11: User interface for the first phase of the LIGHT-Quests data collection task.

B.4.1 Human Demonstration Collection

In order to collect the human completions of quests in the LIGHT environment, I created a game setup where humans could interact with models while playing LIGHT characters in LIGHT settings. I trained a ranking dialogue model on the utterances in the LIGHT dataset.

Using this, players could now assume the role of a LIGHT character and interact with the model. In order to try to control for quality of the quest completions, I used the same ranking model to rank the scores of the player in the dialogues. Players who gave responses that the model ranked as likely candidates would receive more points.

Only after scoring enough cumulative points were players allowed to try completing quests. The quest setup was a slight variation of the conversation setup. First, the player was given one of the collected quest scenarios rather than just a chat setup. Players receiving a quest would be provided with one of the motivations alongside their persona.

In the dialogue that followed, players were given the chance to take action after enough in-character dialogue turns. If the player took the correct action, they were awarded with more points to confirm they completed their given quest.

B.4.2 Examples

I present 3 randomly selected examples of quests and corresponding human demonstrations.

B.4.3 Training and Hyperparameters

Table B.1 gives details on hyperparameters used to train the poly-encoders. Encoders were trained until validation accuracy across all the tasks did not improve for 5 epochs or 24 wall clock hours on a machine with 8 V100 GPUs.

Table B.2 has the hyperparameters used in the RL experiments. Loss coefficients are separated by action and speech types, note that the ratio between the loss coefficients

Setting	You are in the swamp. The swamp is glowing with wonder and color. There are parts that range from dark red to bright yellow. People often visit here to speak with the gods and claim it can be both harmful to those it dislikes and healing to those who it deems worthy. There's a pit of quicksand and a swamp flower here. A witch is here.
Partner:	Witch.
Persona	I grew up in a nearby village, and was exiled when it was found that I had special abilities. My parents were ostracized as well. Since then, I've been on my own, but could never quite let go of my family.
Carrying	Nothing.
Wielding	gold necklace, robe, knife, staff
Self:	Swamp monster.
Persona	I am a swamp monster of the bog. I eat people. I swim around.
Carrying	Nothing.
Wielding	stick, rock

Motivations:		**Timeline:**	
Short	I need some thick foliage to begin construction of my concealed swamp hut.	-2 hours	go to swamp
		-15 min	eat people
		-10 min	follow princess
		Now	get impassable vegetation from pit of quicksand
Mid	I will completely camouflage my swamp hut, so that the King's men won't be able to drive me out even further from the castle.	+15 min	use impassable vegetation with swamp hut
Long	I must live close to the castle, so that I can take the princess away from the evil King.	+1 hours	follow king
		+2 hours	follow princess

What are you doing here witch? GET OUT OF MY SWAMP

I was taken from my family when I was 8 and I need to get out of here! Can you assist me?

Help? HA! I help no one but myself. Speaking of...you look rather plump and tasty witch

Plump?! I'm healthy. I'll tear you up then and make scraps. You watch.

get impassable vegetation from pit of quicksand
You would make a great addition to my stew

It's going to just be gross!

drop impassable vegetation
Get out of my way so I can make my hut. Can't a swamp monster get any peace around here?!

I'll help you, but only so you won't eat me!

That's it, you're coming with me!
get impassable vegetation

I don't trust you. Get off of me!

Table B.1: Hyperparameters used to train all poly-encoders in the supervised experiments. All models have 256 million total parameters. The same trained models were then frozen and used for the RL experiments.

Hyperparameter type	Value
Dictionary Tokenizer	Byte-pair encoding
Num. layers	12
Num. attention heads	12
Feedforward network hidden size	3072
Input length	1024
Embedding size	768
Batch size	32
Dropout ratio	0.1
Poly-n-codes	64
Gradient clip	1.0
Optimizer	Adam
Learning rate	1×10^{-6}

matches the ratio between the sizes of the action spaces. RL experiments were performed on a machine with 8 V100 GPUs for 1 million environment interactions for each actor in a

Setting	This is the hidden workshop of the most powerful wizard in the land. There are ornate tapestries on the walls depicting wizards using their powers and potions in battle. Mordak, the wizard, constructed this powerful workshop after the death of the most famous king, Henry of Silverton. Any who enter here immediately become enchanted with the wizard's power, giving them advanced healing powers. There's a tapestry, a potion, and a tome here. The wizard is here.
Partner:	Wizard.
Persona	I am a wizard who develops my own spells. Most of them aren't particularly effective spells, but I'm curious about all the magical possibilities. People are afraid to participate in my experiments.
Carrying	Nothing.
Self:	Apprentice.
Persona	I am your apprentice. Please tell me what I can help you with. I will cook and serve your meals. I will clean the castle. I can do anything you ask. You have hired me to make your life easier.
Carrying	Nothing.

Motivations:		**Timeline:**	
Short	I need to get the tapestry to clean it.	-2 hours	get hired from wizard
		-15 min	go to secret magician's workshop
		Now	get tapestry
Mid	I need to make this workshop suitable for the wizard.	+5 min	wield tool
		+10 min	hit tapestry
Long	I was hired to keep this place cleaned and in perfect condition for the wizard.	+30 min	put tapestry in wall
		+4 hours	drop tool

Good day Ser Wizard. Your tower is decorated with beautiful tapestries, though their colors appear to be dulled due to dust. May I take it and clean it?

Why not, it is infused isn't it. Just don't be waving it around this room, it might get dangrous

Of course, I will handle it with the utmost care.

How long have you been an apprentice?

get tapestry

3 years Ser. I'm hoping to learn to be a wizard or to become a knight. Or both! Wouldn't that be grand?

How wonderful. What encouraged you to pursue it?

Curiosity mostly. I hope to make the world a better place, and one of the best ways to do that is vanquishing evil

What got you into that occupation then? I was born with affinity for magic so it was my calling.

hug wizard

As I said, curiosity. I am a high born boy, the third son, so I cannot inherit my father's lands. So I must make my mark on the world another way

You are well suited to it and I am sure your parents are proud of you.

batch of 32.

B.4.4 Switch Type Ablations

The second set of results involve ablating having a learned switch that uses the input training data and a hardcoded switch- The learned switch is as described in Chapter 7: it outputs an action every k dialogue utterances; where during training k is chosen to match the ratio of utterances to actions on that particular quest from the human demonstrations, and during testing, k is chosen to match the average action to utterance ratio. The hardcoded switch is where the agent outputs an action chosen every N steps across all quests—here $N = 3$ is the chosen hyperparameter. Table B.3 shows that having a learned switch increases zero-

Setting	You are in the The Queen's Chamber. This is a beautiful room inside of the palace that is decorated with the finest silk and velvet. The color scheme used represents royalty, royal blue, red, green and purple. The walls are covered in gold and in each corner of the room are golden statues of Greek art. The floors are covered in marble, and despite the patterns, shine so brightly you can even see your own reflection in them! There's also a bed big enough to fit five people on! There's two statues, an a bed big, a the finest silk and velvet, an a bed, and a finest silk and velvet here. The butler is here.
Partner:	Butler.
Persona	I serve my masters quietly. I know all the secrets of the elite but will never tell a soul. I have lived in this home since I was 12.
Carrying	Nothing.
Self:	Jester.
Persona	I am the fun guy. I like to entertain others in the village. I am the local jester.
Carrying	Nothing.

Motivations:		**Timeline:**	
Short	I want to hug the butler to cheer him up.	-2 hours	wear Jester's Hat
		-30 min	go Queen's Tower
		-5 min	follow the butler
		Now	hug the butler
Mid	I need to cheer him up because the Queen has just chastised him for dirtying the marble floors.	+5 min	go dining hall
		+10 min	get beef stew
Long	It is my job to cheer up courtiers who are unhappy, and I will lose my home in the kingdom if I don't cheer them up!	+4 hours	give beef stew to butler

Why hello there Butler my man

Hello jester! I'm happy to see you, since I hate my life.

Why so down with the life feels huh

I can't complain (because the king will punish me) everyone wishes they could be the king.

hug butler

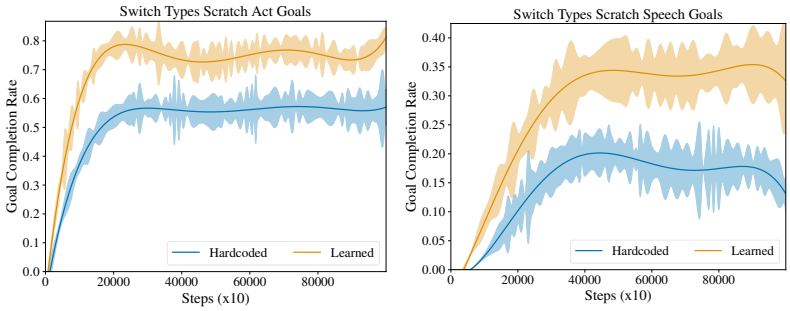
I appreciate the kind words, dear jester.

I'm here for ya. To cheer you up

That is kind of you, not everyone has liked me here, I am the queen's least favorite person.

Well I like you much more than the queen.

shot generalization performance and Figures B.12, B.13 show that having a learned switch improves sample efficiency by enabling the LIGHT agent to reach asymptotic performance in fewer steps in both the Scratch and Adaptive models.



(a) Act Goals (b) Speech Goals

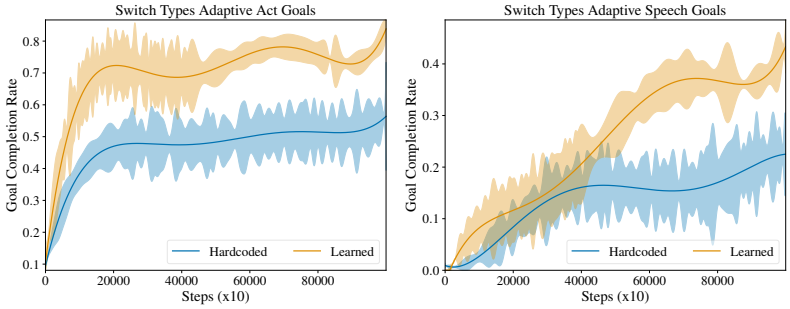
Figure B.12: Switch Types Reward Curves for the Scratch Model averaged over 3 independent runs.

Table B.2: RL experiments hyperparameters. All pre-training encoder hyperparameters are as found earlier in Table B.1.

Hyperparameter type	Value
General	
Discount γ	0.99
Valid Action loss coefficient	10
Action entropy coefficient	0.01
Valid Speech loss coefficient	40
Speech entropy coefficient	0.04
Batch size	32
Gradient clip	1.0
Steps per episode	100
Policy Networks (Actors)	
Num. Layers	3
Feedforward network hidden size	768
GRU hidden size	768
Value Predictor (Critic)	
Num. Layers	2
Feedforward network hidden size	768
Appended Encoder	
Num. layers	3
Num. attention heads	3
Feedforward network hidden size	768

Table B.3: Encoder Type RL Zero-Shot Evaluations averaged over 3 independent runs. Act goals and speech goals are as described in Chapter 7. Standard deviations for all experiments are less than 0.01. The "Act & Speech Goals" column refers to quests where the agent has simultaneously achieved both types of goals within the allotted one episode.

Model	Reinforcement Learning		
	Act Goals	Speech Goals	Act & Speech Goals
Scratch	0.418	0.118	0.103
Hardcoded Switch	0.295	0.0947	0.066
Adaptive	**0.42**	**0.33**	**0.303**
Hardcoded Switch	0.274	0.294	0.236



(a) Act Goals (b) Speech Goals

Figure B.13: Switch Types Reward Curves for the Adaptive Model averaged over 3 independent runs.

B.4.5 Self Act Completion Transcripts

I pick 3 transcripts of the LIGHT agent playing the quests. Each of these transcripts are from the Adaptive model on the test set of human demonstrations during zero-shot evaluations. I pick samples where the agent achieves the *act goal* itself and also simultaneously achieves the *speech goal*. The blue and right aligned text is the LIGHT agent trained with RL and the gray and left aligned text is the partner agent.

Setting	The fishing store is a small one room stone building with wares laid about on tables. One can see fishing poles, wooden buckets with dirt and bait inside, along with some mounted trophies, and a skeleton.
Partner:	Fish seller.
Persona	I sell fish in the kingdom. I usually perch on the high seas. I catch thousands of fish each year.
Carrying	fish
Self:	Mermaid hunter.
Persona	I am a fierce hunter. I bring down mermaids. I spend most of my time on the high seas.
Carrying	crossbow, skeleton, coins
Motivation	I will hit the fish seller to remind them who I am.

I'll get them!!

follow fish seller

hit mermaid hunter

That's it! I am going to eat you. Say your last words.

hug mermaid hunter

Do you see these skeletons I am carrying? They are mermaids I've already devoured!

get skeleton

Get away from my skeletons! The bones are my trophy!

give skeleton to mermaid hunter

It is all your fault my last mermaid got away!

And you are a simpleton, just take this fish. I do not even care about payment.

I DONT WANT YOUR DAMN FISH

get coins from fist

hit fish seller

Setting	The king's bedroom. The walls are tall and stone. They are coated with colorful tapestries showing the kings of years past. A large stone fireplace across from the bed that is large enough to keep the king warm even on the coldest nights. A double thick wooden door with a large lock on each side of the room.
Partner:	Royal dog.
Persona	I am the royal dog, fat, incontinent, and lazy. I eat off my own porcelain plate, when I am not hand fed by the queen. I pee where I like. I stole food from the poor when I was young, but I cannot waddle up to them fast enough these days. I sleep between the royal couple.
Carrying	Nothing.
Self:	Queen.
Persona	I was the daughter of a high ranking nobleman overseas. To make a trade alliance with the King, my parents offered me in marriage. It wasn't my idea, but it has turned out very well. I've produced two living sons as heirs, and the king treats me kindly. I spend my time doing embroidery and talking with my ladies in waiting.
Carrying	Nothing.
Motivation	I want to get to the large stone fireplace.

I am having turbulent thoughts regarding my faith and own morality.

I want to jump into the fireplace.

Don't, lets go to the fireplace and sit.

I go to fireplace.

get large stone fireplace

Setting	The sheriff's office is a small building on the southern edge of the farm. There is a desk with decrees from the king and other papers strewn about. Hanging on the wall are swords and other weapons. The room is small and cluttered, with various books and other things strewn about. There is a guard here.
Partner:	Lawyer.
Persona	I am a very good lawyer. I always win in my cases. I would like to work as a district attorney.
Carrying	Nothing.
Self:	Sheriff.
Persona	I keep our town safe. I make sure to check the caravans for bandits. If the King needs anything extra hands in the kingdom he can count on me.
Carrying	Nothing.
Motivation	I want to remove the shield and hit the lawyer with it.

Next time I'll hit you with this shield! Tell the truth, it was you that went after the king! I don't buy the frame story. No matter how good a lawyer you are. You aren't fooling me!

get shield

remove knife

Hahahahaha. I'm off, don't like to fight! You are a fool sheriff. I can run faster than you can ever imagine.

follow lawyer

This is my shield now! Goodbye! I'm off.

Stop!!

put shield on large and well stocked wine rack

get shield from large and well stocked wine rack

hug sheriff

hit lawyer

Goodbye.

Alright, lawyer. You may go now.

B.4.6 Partner Act Completion Transcripts

Similarly to the previous section, these transcripts are from the Adaptive model on the test set of human demonstrations during zero-shot evaluations. I pick samples where the *act goal* is achieved via the partner and the agent simultaneously achieves the *speech goal* itself. The blue and right aligned text is the LIGHT agent trained with RL and the gray and left aligned text is the partner agent.

Setting	Up at the top of the tower, among the gray clouds, is the prison room. The room is round, with a circumference of about 15 feet. The walls are barren and made of stone; only one barred window gives a view out onto the vast landscape. A small bed and a dirty wooden bucket are the only decorations. The desolate fairy spends day and night laying on that small bed, wondering where things went wrong.
Partner:	Fairy.
Persona	I'm a fairy that lives in the forest. I like to fly around with my siblings and play games. I try to hide from humans. If they find me, I'll grant their wishes.
Carrying	Nothing.
Self:	Witch.
Persona	Halloween is my favorite time of the year. My familiar is a cat named Helix. I have a spell book that is 300 years old. I search the forest for herbs and berries for my potions.
Carrying	Nothing.
Motivation	I need the dirty wooden bucket to make a potion.

Take the doll

hug fairy

I will take this doll back to the palace with me.

You do you fairy.

hug witch

You know whats going on w/ the princess?

She just disappeared suddenly late at night without a sound heard

Anyway, I will make this potion and drink it, but I do not think it will help me.

I will drink the potion but why make it.

Because we're in a prison tower. Duh. Also give me that bucket or I'll feed you to my cat!

give dirty wooden bucket to witch

Setting	The throne room. Very luxurious, with details of 12 carat gold and diamonds with a very comfortable cushions where the king usually takes a break.
Partner:	Powerful but aged wizard.
Persona	I am the official wizard to the king. I give him advice on how to run the kingdom. I enjoy and love using magic in my everyday life. I'm getting old now but I still have a zest for life. I really enjoy doing magic for the local peasants in the village.
Wielding	robe, staff
Self:	King's trusted adviser.
Persona	I am the king's trusted adviser. I hold the discretion of the king. I guard the secrets and strategies that make my kingdom thrive. I live a lonely life because others cannot be trusted.
Wielding	slippers, robes
Motivation	I need to bring the king's pet to him

Well, I was doing what you told me to do. I was going to eat him but there was hardly any meat on him.

I killed him before he could grow.

Carry me to pond.

Yes, milord, I shall do as you command.

remove slippers

Where is King Fulmer's pet?

Is he not in his bed?

Yes, I guess you need to bring him in.

follow powerful but aged wizard

I want to become the new head wizard.

Nope he just probably throw you overboard.

The wizard is evil? Dang it. Help me.

You're not the lion, I need to get the lion out.

give large pet lion to king's trusted adviser

Setting	The Tent. Shabby, torn fabric, stained by weeks of travel and bleached from the harsh sun. Long, thin sticks held the fabric up so it formed a triangle with the earth. The sticks were buried deep within the shifting orange sand to hold off the blistering wind.
Partner:	Person.
Persona	I'm just a guy. I work at the saw mill every day. I'm a good villager.
Carrying	Nothing.
Self:	Military Commander.
Persona	I am the military commander of the village. It is my duty to train and lead our soldiers into battle. I am a stern commander and I expect the best from my troops.
Carrying	sword
Motivation	I need to get sand to use as a tactical planning prop.

Just ensure the home front is properly protected.

hug person

Perfect! God speed commander.

Yes. I need to prepare.

Now. I heard the enemy is coming.

pick up stick

get stick

Where's the sand?

give sand to military commander

B.4.7 Curriculum Learning

This section provides the hyperparamters for the models used in the procedural LIGHT environment generation pipeline. Agents trained on these curriculums have the same hyperparameters as referenced in Chapter 7.

Table B.4: Hyperparameters used to train transformer/ranker model to retrieve objects for generating the LIGHT world. The same trained models were then frozen and used for further RL experiments.

Hyperparameter type	Value
Num. layers	2
Num. attention heads	2
Embedding size	300
Dropout ratio	0.0
Gradient clip	0.1
Optimizer	Adam
Learning rate	1×10^{-4}

Table B.5: Hyperparameters used to train starspace model to retrieve character for generating the LIGHT world. The same trained models were then frozen and used for further RL experiments.

Hyperparameter type	Value
Embedding size	128
Embedding norm	10
Dropout ratio	0.0
Gradient clip	0.1
Optimizer	SGD
Learning rate	0.1

Table B.6: Hyperparameters used to train BART model for generating short motivations. The same trained models were then frozen and used for further RL experiments.

Hyperparameter type	Value
Num. encoder layers	12
Num. decoder layers	12
Num. attention heads	16
Batchsize	4
Activation	gelu
Beam size	1
Beam decay	30
Beam length penalty	0.65
Num. attention heads	2
Embedding size	1024
Dropout ratio	0.1
Gradient clip	0.1
Optimizer	SGD
Learning rate	1×10^{-4}

Table B.7: Hyperparameters used to train BART model for generating goals. The same trained models were then frozen and used for further RL experiments.

Hyperparameter type	Value
Num. encoder layers	12
Num. decoder layers	12
Num. attention heads	16
Batchsize	4
Activation	gelu
Beam size	1
Beam decay	30
Beam length penalty	0.65
Num. attention heads	2
Embedding size	1024
Dropout ratio	0.1
Gradient clip	0.1
Optimizer	SGD
Learning rate	1×10^{-4}

REFERENCES

[1] J. Feldman and S. Narayanan, "Embodied meaning in a neural theory of language," *Brain and language*, vol. 89, pp. 385–92, 2004.

[2] L. W. Barsalou, "Grounded cognition," *Annual Review of Psychology*, vol. 59, no. 1, pp. 617–645, 2008, PMID: 17705682. eprint: https://doi.org/10.1146/annurev.psych.59.103006.093639.

[3] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *Journal of Artificial Intelligence Research*, vol. 47, pp. 253–279, 2013.

[4] M. Johnson, K. Hofmann, T. Hutton, and D. Bignell, "The malmo platform for artificial intelligence experimentation," in *IJCAI*, ser. IJCAI'16, New York, New York, USA: AAAI Press, 2016, pp. 4246–4247, ISBN: 978-1-57735-770-4.

[5] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, D. Gordon, Y. Zhu, A. Gupta, and A. Farhadi, "AI2-THOR: An Interactive 3D Environment for Visual AI," *arXiv*, 2017.

[6] Manolis Savva*, Abhishek Kadian*, Oleksandr Maksymets*, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra, "Habitat: A Platform for Embodied AI Research," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[7] R. A. Brooks, "Intelligence without representation," *Artificial intelligence*, vol. 47, no. 1-3, pp. 139–159, 1991.

[8] T. Mikolov, A. Joulin, and M. Baroni, "A roadmap towards machine intelligence," in *International Conference on Intelligent Text Processing and Computational Linguistics*, Springer, 2016, pp. 29–61.

[9] J. Gauthier and I. Mordatch, "A paradigm for situated and goal-driven language learning," *arXiv preprint arXiv:1610.03585*, 2016.

[10] B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman, "Building machines that learn and think like people," *Behavioral and brain sciences*, vol. 40, 2017.

[11] M. O. Riedl and V. Bulitko, "Interactive narrative: An intelligent systems approach," *Ai Magazine*, vol. 34, no. 1, pp. 67–67, 2013.

[12] T. Anderson, M. Blank, B. Daniels, and D. Lebling, *Zork*, http://ifdb.tads.org/viewgame?id=4gxk83ja4twckm6j, 1979.

[13] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 1st. Cambridge, MA, USA: MIT Press, 1998, ISBN: 0262193981.

[14] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu, "Reinforcement learning with unsupervised auxiliary tasks," *CoRR*, vol. abs/1611.05397, 2016. arXiv: 1611.05397.

[15] OpenAI, M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J. W. Pachocki, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba, "Learning dexterous in-hand manipulation," *CoRR*, vol. abs/1808.00177, 2018. arXiv: 1808.00177.

[16] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005, ISBN: 0262201623.

[17] P. Ammanabrolu and M. O. Riedl, "Playing text-adventure games with graph-based deep reinforcement learning," in *Proceedings of 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019*, Minneapolis, Minnesota, 2019.

[18] K. Murugesan, M. Atzeni, P. Shukla, M. Sachan, P. Kapanipathi, and K. Talamadupula, "Enhancing text-based reinforcement learning agents with commonsense knowledge," *arXiv preprint arXiv:2005.00811*, 2020.

[19] A. Adhikari, X. Yuan, M.-A. Côté, M. Zelinka, M.-A. Rondeau, R. Laroche, P. Poupart, J. Tang, A. Trischler, and W. Hamilton, "Learning dynamic belief graphs to generalize on text-based games," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[20] P. Ammanabrolu and M. Hausknecht, "Graph Constrained Reinforcement Learning for Natural Language Action Spaces," in *International Conference on Learning Representations*, 2020.

[21] T. Zahavy, M. Haroush, N. Merlis, D. J. Mankowitz, and S. Mannor, "Learn what not to learn: Action elimination with deep reinforcement learning," in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., Curran Associates, Inc., 2018, pp. 3562–3573.

[22] X. Yuan, M. Côté, A. Sordoni, R. Laroche, R. T. des Combes, M. J. Hausknecht, and A. Trischler, "Counting to explore and generalize in text-based games," *CoRR*, vol. abs/1806.11525, 2018. arXiv: 1806.11525.

[23] X. Yin and J. May, "Comprehensible context-driven text game playing," *CoRR*, vol. abs/1905.02265, 2019.

[24] M. Hausknecht, P. Ammanabrolu, M.-A. Côté, and X. Yuan, "Interactive fiction games: A colossal adventure," in *Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

[25] Y. Jang, S. Seo, J. Lee, and K.-E. Kim, "Monte-carlo planning and learning with language action value estimates," in *International Conference on Learning Representations*, 2021.

[26] P. Ammanabrolu, E. Tien, M. Hausknecht, and M. O. Riedl, "How to avoid being eaten by a grue: Structured exploration strategies for textual worlds," *arXiv preprint arXiv:2006.07409*, 2020.

[27] A. Ecoffet, J. Huizinga, J. Lehman, K. O. Stanley, and J. Clune, "First return, then explore," *Nature*, vol. 590, no. 7847, pp. 580–586, Feb. 1, 2021.

[28] A. Madotto, M. Namazifar, J. Huizinga, P. Molino, A. Ecoffet, H. Zheng, A. Papangelis, D. Yu, C. Khatri, and G. Tur, "Exploration based language learning for text-based games," *CoRR*, vol. abs/2001.08868, 2020.

[29] J. Urbanek, A. Fan, S. Karamcheti, S. Jain, S. Humeau, E. Dinan, T. Rocktäschel, D. Kiela, A. Szlam, and J. Weston, "Learning to speak and act in a fantasy text adventure game," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

[30] P. Ammanabrolu, J. Urbanek, M. Li, A. Szlam, T. Rocktäschel, and J. Weston, "How to motivate your dragon: Teaching goal-driven agents to speak and act in fantasy worlds," in *Proceedings of 2021 Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021*, 2021.

[31] B. Boyd, "The evolution of stories: From mimesis to language, from fact to fiction," *WIREs Cognitive Science*, vol. 9, no. 1, e1444, 2018. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/wcs.1444.

[32] P. Ammanabrolu, W. Broniec, A. Mueller, J. Paul, and M. O. Riedl, "Toward automated quest generation in text-adventure games," in *International Conference on Computational Creativity (ICCC)*, 2020.

[33] P. Ammanabrolu, W. Cheung, D. Tu, W. Broniec, and M. O. Riedl, "Bringing stories alive: Generating interactive fiction worlds," in *Proceedings of the Sixteenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE-20)*, 2020.

[34] M.-A. Côté, A. Kádár, X. Yuan, B. Kybartas, T. Barnes, E. Fine, J. Moore, M. Hausknecht, L. E. Asri, M. Adada, W. Tay, and A. Trischler, "Textworld: A learning environment for text-based games," *CoRR*, vol. abs/1806.11532, 2018.

[35] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, *Openai gym*, 2016. eprint: arXiv:1606.01540.

[36] R. Coulom, "Efficient selectivity and backup operators in monte-carlo tree search," in *Computers and Games*, H. J. van den Herik, P. Ciancarini, and H. H. L. M. J. Donkers, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 72–83, ISBN: 978-3-540-75538-8.

[37] C. Resnick, R. Raileanu, S. Kapoor, A. Peysakhovich, K. Cho, and J. Bruna, "Backplay: "man muss immer umkehren"," *CoRR*, vol. abs/1807.06919, 2018. arXiv: 1807.06919.

[38] S. Humeau, K. Shuster, M.-A. Lachaux, and J. Weston, "Poly-encoders: Architectures and pre-training strategies for fast and accurate multi-sentence scoring," in *International Conference on Learning Representations*, 2020.

[39] M. Horsfall and A. Oikonomou, "A study of how different game play aspects can affect the popularity of role-playing video games," in *2011 16th International Conference on Computer Games (CGAMES)*, IEEE, 2011, pp. 63–69.

[40] M. Shridhar, X. Yuan, M.-A. Cote, Y. Bisk, A. Trischler, and M. Hausknecht, "{alfw}orld: Aligning text and embodied environments for interactive learning," in *International Conference on Learning Representations*, 2021.

[41] K. Murugesan, M. Atzeni, P. Kapanipathi, P. Shukla, S. Kumaravel, G. Tesauro, K. Talamadupula, M. Sachan, and M. Campbell, "Text-based RL Agents with Commonsense Knowledge: New Challenges, Environments and Baselines," in *Thirty Fifth AAAI Conference on Artificial Intelligence*, 2021.

[42] R. Tamari, F. Bai, A. Ritter, and G. Stanovsky, "Process-level representation of scientific protocols with interactive annotation," *arXiv preprint arXiv:2101.10244*, 2021.

[43] J. He, J. Chen, X. He, J. Gao, L. Li, L. Deng, and M. Ostendorf, "Deep reinforcement learning with a natural language action space," in *ACL*, 2016.

[44] M. Zelinka, "Using reinforcement learning to learn how to play text-based games," *CoRR*, vol. abs/1801.01999, 2018. arXiv: 1801.01999.

[45] K. Narasimhan, T. D. Kulkarni, and R. Barzilay, "Language understanding for text-based games using deep reinforcement learning," in *EMNLP*, 2015, pp. 1–11.

[46] N. Fulda, D. Ricks, B. Murdoch, and D. Wingate, "What can you do with a rock? affordance extraction via word embeddings," in *IJCAI*, 2017, pp. 1039–1045.

[47] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *CoRR*, vol. abs/1301.3781, 2013. arXiv: 1301.3781.

[48] X. Yuan, M.-A. Côté, J. Fu, Z. Lin, C. Pal, Y. Bengio, and A. Trischler, "Interactive language learning by question answering," in *EMNLP*, 2019.

[49] C. Sautier, D. J. Agravante, and M. Tatsubori, "State Prediction in TextWorld with a Predicate-Logic Pointer Network Architecture," in *In Workshop on Knowledge-based Reinforcment Learning at IJCAI-20*, 2020.

[50] S. Dambekodi, S. Frazier, P. Ammanabrolu, and M. O. Riedl, "Playing text-based games with common sense," *arXiv preprint arXiv:2012.02757*, 2020.

[51] G. Konidaris and A. G. Barto, "Building Portable Options: Skill Transfer in Reinforcement Learning," in *IJCAI*, 2007.

[52] G. Konidaris, I. Scheidwasser, and A. G. Barto, "Transfer in Reinforcement Learning via Shared Features," *The Journal of Machine Learning Research*, vol. 13, pp. 1333–1371, 2012.

[53] Y. Liu and P. Stone, "Value-Function-Based Transfer for Reinforcement Learning Using Structure Mapping," in *AAAI*, 2006.

[54] M. E. Taylor, N. K. Jong, and P. Stone, "Transferring Instances for Model-Based Reinforcement Learning," in *ECML/PKDD*, 2008.

[55] T. T. Nguyen, T. Silander, and T.-Y. Leong, "Transferring Expectations in Model-based Reinforcement Learning," in *NIPS*, 2012.

[56] M. Gasic, C. Breslin, M. Henderson, D. Kim, M. Szummer, B. Thomson, P. Tsiakoulis, and S. J. Young, "POMDP-based dialogue manager adaptation to extended domains," in *SIGDIAL Conference*, 2013.

[57] Z. Wang, T.-H. Wen, P.-h. Su, and Y. Stylianou, "Learning Domain-Independent Dialogue Policies via Ontology Parameterisation," in *SIGDIAL Conference*, 2015.

[58] G. Joshi and G. Chowdhary, "Cross-Domain Transfer in Reinforcement Learning Using Target Apprentice," in *Proceedings of the International Conference on Robotics and Automation*, 2018, pp. 7525–7532.

[59] K. Narasimhan, R. Barzilay, and T. Jaakkola, "Deep Transfer in Reinforcement Learning by Language Grounding," *Journal of Artificial Intelligence Research*, vol. 63, 2017.

[60] E. Parisotto, J. Ba, and R. R. Salakhutdinov, "Actor-Mimic: Deep Multitask and Transfer Reinforcement Learning," *CoRR*, vol. abs/1511.06342, 2016.

[61] Yin H. and S. J. Pan, "Knowledge transfer for deep reinforcement learning with hierarchical experience replay," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, ser. AAAI'17, AAAI Press, 2017, pp. 1640–1646.

[62] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive Neural Networks," *CoRR*, vol. abs/1606.04671, 2016.

[63] J. Rajendran, A. S. Lakshminarayanan, M. M. Khapra, P Prasanna, and B. Ravindran, "Attend, Adapt and Transfer: Attentive Deep Architecture for Adaptive Transfer from multiple sources in the same domain," in *ICLR*, 2017.

[64] V. Jain, W. Fedus, H. Larochelle, D. Precup, and M. G. Bellemare, "Algorithmic improvements for deep reinforcement learning applied to interactive fiction," *CoRR*, vol. abs/1911.12511, 2019.

[65] M. G. Bellemare, G. Ostrovski, A. Guez, P. Thomas, and R. Munos, "Increasing the action gap: New operators for reinforcement learning," in *AAAI Conference on Artificial Intelligence*, 2016.

[66] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 41–48.

[67] J. Schmidhuber, "Powerplay: Training an increasingly general problem solver by continually searching for the simplest still unsolvable problem," *Frontiers in psychology*, vol. 4, p. 313, 2013.

[68] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone, "Curriculum learning for reinforcement learning domains: A framework and survey," *Journal of Machine Learning Research*, vol. 21, no. 181, pp. 1–50, 2020.

[69] S. Sukhbaatar, Z. Lin, I. Kostrikov, G. Synnaeve, A. Szlam, and R. Fergus, "Intrinsic motivation and automatic curricula via asymmetric self-play," in *International Conference on Learning Representations*, 2018.

[70] S. Racaniere, A. K. Lampinen, A. Santoro, D. P. Reichert, V. Firoiu, and T. P. Lillicrap, "Automated curricula through setter-solver interactions," *arXiv preprint arXiv:1909.12892*, 2019.

[71] A. Campero, R. Raileanu, H. Kuttler, J. B. Tenenbaum, T. Rocktäschel, and E. Grefenstette, "Learning with {amig}o: Adversarially motivated intrinsic goals," in *International Conference on Learning Representations*, 2021.

[72] A. Graves, M. G. Bellemare, J. Menick, R. Munos, and K. Kavukcuoglu, "Automated curriculum learning for neural networks," in *International Conference on Machine Learning*, 2017, pp. 1311–1320.

[73] R. Portelas, C. Colas, K. Hofmann, and P.-Y. Oudeyer, "Teacher algorithms for curriculum learning of deep rl in continuously parameterized environments," in *Conference on Robot Learning*, PMLR, 2020, pp. 835–853.

[74] M. Henderson, B. Thomson, and J. D. Williams, "The second dialog state tracking challenge," in *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, 2014, pp. 263–272.

[75] L. El Asri, H. Schulz, S. Sharma, J. Zumer, J. Harris, E. Fine, R. Mehrotra, and K. Suleman, "Frames: A corpus for adding memory to goal-oriented dialogue systems," in *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, Saarbrücken, Germany: Association for Computational Linguistics, 2017, pp. 207–219.

[76] S. P. Singh, M. J. Kearns, D. J. Litman, and M. A. Walker, "Reinforcement learning for spoken dialogue systems," in *Advances in Neural Information Processing Systems*, 2000, pp. 956–962.

[77] O. Pietquin, M. Geist, S. Chandramohan, and H. Frezza-Buet, "Sample-efficient batch reinforcement learning for dialogue management optimization," *ACM Transactions on Speech and Language Processing (TSLP)*, vol. 7, no. 3, p. 7, 2011.

[78] M. Fatemi, L. E. Asri, H. Schulz, J. He, and K. Suleman, "Policy networks with two-stage training for dialogue systems," *arXiv preprint arXiv:1606.03152*, 2016.

[79] J. Li, W. Monroe, A. Ritter, M. Galley, J. Gao, and D. Jurafsky, "Deep reinforcement learning for dialogue generation," *CoRR*, vol. abs/1606.01541, 2016. arXiv: 1606.01541.

[80] D. Yarats and M. Lewis, "Hierarchical text generation and planning for strategic dialogue," *arXiv preprint arXiv:1712.05846*, 2017.

[81] M. Lewis, D. Yarats, Y. N. Dauphin, D. Parikh, and D. Batra, "Deal or no deal? end-to-end learning for negotiation dialogues," *arXiv preprint arXiv:1706.05125*, 2017.

[82] T. Trabasso and P. van den Broek, "Causal thinking and the representation of narrative events," *Journal of Memory and Language*, vol. 24, pp. 612–630, 1985.

[83] A. Graesser, K. L. Lang, and R. M. Roberts, "Question answering in the context of stories," *Journal of Experimental Psychology: General*, vol. 120, no. 3, pp. 254–277, 1991.

[84] A. Bosselut, H. Rashkin, M. Sap, C. Malaviya, A. Çelikyilmaz, and Y. Choi, "Comet: Commonsense transformers for automatic knowledge graph construction," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019.

[85] J. Guan, F. Huang, Z. Zhao, X. Zhu, and M. Huang, "A knowledge-enhanced pretraining model for commonsense story generation," *Transactions of the Association for Computational Linguistics*, 2020.

[86] M. Sap, R. Le Bras, E. Allaway, C. Bhagavatula, N. Lourie, H. Rashkin, B. Roof, N. A. Smith, and Y. Choi, "Atomic: An atlas of machine commonsense for if-then reasoning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 3027–3035.

[87] R. Speer and C. Havasi, "Representing general relational knowledge in conceptnet 5," in *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC)*, 2012, ISBN: 978-2-9517408-7-7.

[88] P. Ammanabrolu and M. Riedl, "Transfer in deep reinforcement learning using knowledge graphs," in *Proceedings of the Thirteenth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-13) at EMNLP*, 2019.

[89] J. Luketina, N. Nardelli, G. Farquhar, J. Foerster, J. Andreas, E. Grefenstette, S. Whiteson, and T. Rocktäschel, "A survey of reinforcement learning informed by natural language," *arXiv preprint arXiv:1906.03926*, 2019.

[90] M. MacMahon, B. Stankiewicz, and B. Kuipers, "Walk the talk: Connecting language, knowledge, and action in route instructions," in *AAAI*, 2006.

[91] T. Kollar, S. Tellex, D. Roy, and N. Roy, "Toward understanding natural language directions," in *2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, IEEE, 2010, pp. 259–266.

[92] Y. Bisk, D. Yuret, and D. Marcu, "Natural language communication with robots," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016, pp. 751–761.

[93] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. Reid, S. Gould, and A. van den Hengel, "Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3674–3683.

[94] A. Das, S. Kottur, J. M. Moura, S. Lee, and D. Batra, "Learning cooperative visual dialog agents with deep reinforcement learning," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2951–2960.

[95] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[96] D. Ha and J. Schmidhuber, "Recurrent world models facilitate policy evolution," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31, Curran Associates, Inc., 2018.

[97] J. Oh, X. Guo, H. Lee, R. L. Lewis, and S. Singh, "Action-Conditional Video Prediction using Deep Networks in Atari Games," in *Advances in Neural Information Processing Systems 28*, C Cortes, N. D. Lawrence, D. D. Lee, M Sugiyama, R Garnett, and R Garnett, Eds., Curran Associates, Inc., 2015, pp. 2845–2853.

[98] T. Kipf, E. van der Pol, and M. Welling, "Contrastive learning of structured world models," in *International Conference on Learning Representations*, 2020.

[99] M. Guzdial, B. Harrison, B. Li, and M. Riedl, "Crowdsourcing Open Interactive Narrative.," in *FDG*, 2015.

[100] M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller, "Embed to control: A locally linear latent dynamics model for control from raw images," in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28, Curran Associates, Inc., 2015.

[101] N. Wahlström, T. B. Schön, and M. P. Deisenroth, "From pixels to torques: Policy learning with deep dynamical models," *arXiv preprint arXiv:1502.02251*, 2015.

[102] J. Permar and B. Magerko, "A Conceptual Blending Approach to the Generation of Cognitive Scripts for Interactive Narrative," in *9th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE-13)*, 2013.

[103] T. Veale, D. O'donoghue, and M. T. Keane, "Computation and Blending," *Cognitive Linguistics*, vol. 11(3/4), pp. 253–282, 2000.

[104] A. Zook, B. Magerko, and M. Riedl, "Formally Modeling Pretend Object Play," in *Proceedings of the 8th ACM Conference on Creativity and Cognition*, New York, NY, USA: ACM, 2011, pp. 147–156, ISBN: 978-1-4503-0820-5.

[105] B. Magerko and B. J. O'Neill, "Formal Models of Western Films for Interactive Narrative Technologies," 2012.

[106] B. Li, S. Lee-Urban, D. S. Appling, and M. O. Riedl, "Crowdsourcing Narrative Intelligence," in *Advances in Cognitive Systems*, vol. 1, 2012, pp. 1–18.

[107] L. J. Martin, P. Ammanabrolu, X. Wang, S. Singh, B. Harrison, M. Dhuliawala, P. Tambwekar, A. Mehta, R. Arora, N. Dass, C. Purdy, and M. O. Riedl, "Improvisational Storytelling Agents," in *Workshop on Machine Learning for Creativity and Design (NeurIPS 2017)*, Long Beach, CA, 2017.

[108] S. Giannatos, M. J. Nelson, Y.-G. Cheong, and G. N. Yannakakis, "Suggesting New Plot Elements for an Interactive Story," in *In Workshop on Intellignet Narrative Technologies (INT'11)*, 2011.

[109] K Hartsook, A Zook, S Das, and M. O. Riedl, "Toward supporting stories with procedurally generated game worlds," in *2011 IEEE Conference on Computational Intelligence and Games (CIG'11)*, 2011, pp. 297–304.

[110] R. Tamari, H. Shindo, D. Shahaf, and Y. Matsumoto, "Playing by the Book: An Interactive Game Approach for Action Graph Extraction from Text," in *Proceedings of the Workshop on Extracting Structured Knowledge from Scientific Publications*, Minneapolis, Minnesota: Association for Computational Linguistics, 2019, pp. 62–71.

[111] A. Fan, J. Urbanek, P. Ringshia, E. Dinan, E. Qian, S. Karamcheti, S. Prabhumoye, D. Kiela, T. Rocktaschel, A. Szlam, and Others, "Generating Interactive Worlds with Text," in *In proceedings of the Thirty-Third AAAI Conference on AI (AAAI-19)*, 2019.

[112] D. Jancke, "Orientation formed by a spot's trajectory: A two-dimensional population approach in primary visual cortex," *Journal of Neuroscience*, vol. 20, no. 14, RC86–RC86, 2000.

[113] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, *et al.*, "Mastering atari, go, chess and shogi by planning with a learned model," *arXiv preprint arXiv:1911.08265*, 2019.

[114] J. Guan, Y. Wang, and M. Huang, "Story Ending Generation with Incremental Encoding and Commonsense Knowledge," *arXiv:1808.10113v1*, 2018.

[115] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," *International Conference on Learning Representations (ICLR)*, 2018.

[116] G. A. Miller, "WordNet: A Lexical Database for English," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[117] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 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)*, Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186.

[118] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language Models are Unsupervised Multitask Learners," 2019.

[119] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'14, Cambridge, MA, USA: MIT Press, 2014, 3104–3112.

[120] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, "Deep sets," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30, Curran Associates, Inc., 2017.

[121] G. Angeli, J. Premkumar, M. Jose, and C. D. Manning, "Leveraging Linguistic Structure For Open Domain Information Extraction," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2015.

[122] P. Rajpurkar, R. Jia, and P. Liang, "Know what you don't know: Unanswerable questions for SQuAD," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Melbourne, Australia: Association for Computational Linguistics, 2018, pp. 784–789.

[123] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," in *International Conference on Learning Representations*, 2020.

[124] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online: Association for Computational Linguistics, 2020, pp. 7871–7880.

[125] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: A method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, 2002, pp. 311–318.

[126] S. Yao, R. Rao, M. Hausknecht, and K. Narasimhan, "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 (EMNLP)*, Online: Association for Computational Linguistics, 2020, pp. 8736–8754.

[127] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, 1992.

[128] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 2018, ISBN: 0262193981. arXiv: 1507.04296.

[129] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv:1409.0473*, 2014.

[130] J. He, J. Chen, X. He, J. Gao, L. Li, L. Deng, and M. Ostendorf, "Deep Reinforcement Learning with a Natural Language Action Space," in *Association for Computational Linguistics (ACL)*, 2016. arXiv: 1511.04636.

[131] L.-J. Lin, "Reinforcement learning for robots using neural networks," PhD thesis, Carnegie Mellon University, 1993, ISBN: 0739-0572.

[132] A. W. Moore and C. G. Atkeson, "Prioritized Sweeping: Reinforcement Learning with Less Data and Less Time," *Machine Learning*, 1993.

[133] T. Kudo, "Subword regularization: Improving neural network translation models with multiple subword candidates," *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018.

[134] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, 2016, pp. 1928–1937.

[135] X. Yin and J. May, "Learn how to cook a new recipe in a new house: Using map familiarization, curriculum learning, and common sense to learn families of text-based adventure games," *arXiv preprint arXiv:1908.04777*, 2019.

[136] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999.

[137] M. Stolle and D. Precup, "Learning options in reinforcement learning," in *Proceedings of the 5th International Symposium on Abstraction, Reformulation and Approximation*, Berlin, Heidelberg: Springer-Verlag, 2002, 212–223, ISBN: 3540439412.

[138] A. McGovern and A. G. Barto, "Automatic discovery of subgoals in reinforcement learning using diverse density," 2001.

[139] B. McCann, N. S. Keskar, C. Xiong, and R. Socher, "The Natural Language Decathlon : Multitask Learning as Question Answering," *arXiv:1806.08730*, 2017.

[140] D. Chen, A. Fisch, J. Weston, and A. Bordes, "Reading Wikipedia to answer open-domain questions," in *Association for Computational Linguistics (ACL)*, 2017.

[141] M. E. Taylor and P. Stone, "Transfer Learning for Reinforcement Learning Domains: A Survey," *Journal of Machine Learning Research*, vol. 10, pp. 1633–1685, 2009.

[142] D. Griffith, *Frotz: Infocom-style interactive fiction player for unix and dos*, https://gitlab.com/DavidGriffith/frotz, 2018.

[143] J. Baumgartner, S. Zannettou, B. Keegan, M. Squire, and J. Blackburn, "The pushshift reddit dataset," in *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 14, 2020, pp. 830–839.

[144] T. Degris and O. Sigaud, "Factored markov decision processes," *Markov Decision Processes in Artificial Intelligence*, pp. 99–126, 2013.

[145] S. Prabhumoye, M. Li, J. Urbanek, E. Dinan, D. Kiela, J. Weston, and A. Szlam, "I love your chain mail! making knights smile in a fantasy game world: Open-domain goal-orientated dialogue agents," *arXiv preprint arXiv:2002.02878*, 2020.

[146] J. Lee, K. Cho, and D. Kiela, "Countering language drift via visual grounding," *arXiv preprint arXiv:1909.04499*, 2019.

[147] T. Sellam, D. Das, and A. Parikh, "BLEURT: Learning robust metrics for text generation," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online: Association for Computational Linguistics, 2020, pp. 7881–7892.

[148] Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge graph embedding: A survey of approaches and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 12, pp. 2724–2743, 2017.

[149] S. Roller, E. Dinan, N. Goyal, D. Ju, M. Williamson, Y. Liu, J. Xu, M. Ott, K. Shuster, E. M. Smith, *et al.*, "Recipes for building an open-domain chatbot," *arXiv preprint arXiv:2004.13637*, 2020.

[150] Y. Yang, S. Yuan, D. Cer, S.-Y. Kong, N. Constant, P. Pilar, H. Ge, Y.-H. Sung, B. Strope, and R. Kurzweil, "Learning semantic textual similarity from conversations," *arXiv preprint arXiv:1804.07754*, 2018.

[151] P.-E. Mazaré, S. Humeau, M. Raison, and A. Bordes, "Training millions of personalized dialogue agents," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium: Association for Computational Linguistics, 2018, pp. 2775–2779.

[152] V. Krakovna, J. Uesato, V. Mikulik, M. Rahtz, T. Everitt, R. Kumar, Z. Kenton, J. Leike, and S. Legg, "Specification gaming: The flip side of ai ingenuity," *DeepMind Blog*, 2020.

[153] J. Phang, T. Févry, and S. R. Bowman, "Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks," *arXiv preprint arXiv:1811.01088*, 2018.

[154] S. Gururangan, A. Marasović, S. Swayamdipta, K. Lo, I. Beltagy, D. Downey, and N. A. Smith, "Don't stop pretraining: Adapt language models to domains and tasks," *arXiv preprint arXiv:2004.10964*, 2020.

[155] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations," *arXiv e-prints*, arXiv:1909.11942, 2019. arXiv: 1909.11942 [cs.CL].

[156] M. T. Ribeiro, C. Guestrin, and S. Singh, "Are red roses red? evaluating consistency of question-answering models," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 6174–6184.

[157] S. Saha and Mausam, "Open information extraction from conjunctive sentences," in *Proceedings of the 27th International Conference on Computational Linguistics*, 2018, pp. 2288–2299.

[158] H. Pal and Mausam, "Demonyms and compound relational nouns in nominal open IE," in *Proceedings of the 5th Workshop on Automated Knowledge Base Construction*, 2016, pp. 35–39.

[159] J. Christensen, S. Soderland, O. Etzioni, and Mausam, "An analysis of open information extraction based on semantic role labeling," in *Proceedings of the sixth international conference on Knowledge capture*, ACM, 2011, pp. 113–120.

[160] N. Shirish Keskar, B. McCann, L. R. Varshney, C. Xiong, and R. Socher, "CTRL: A Conditional Transformer Language Model for Controllable Generation," *arXiv e-prints*, arXiv:1909.05858, arXiv:1909.05858, Sep. 2019. arXiv: 1909.05858 [cs.CL].

[161] D. Ippolito, D. Grangier, C. Callison-Burch, and D. Eck, "Unsupervised Hierarchical Story Infilling," in *Proceedings of the First Workshop on Narrative Understanding*, 2019, pp. 37–43.

[162] C. Donahue, M. Lee, and P. Liang, "Enabling language models to fill in the blanks," in *Association for Computational Linguistics*, 2020.

[163] K. Clark and C. D. Manning, "Deep reinforcement learning for mention-ranking coreference models," in *EMNLP*, 2016.

[164] P. Ammanabrolu, E. Tien, W. Cheung, Z. Luo, W. Ma, L. J. Martin, and M. O. Riedl, "Story Realization: Expanding Plot Events into Sentences," in *AAAI Conference on Artificial Intelligence*, 2020, p. 8.

[165] A. See, A. Pappu, R. Saxena, A. Yerukola, and C. D. Manning, "Do Massively Pretrained Language Models Make Better Storytellers?," 2019. arXiv: 1909.10705.

[166] F. Petroni, T. Rocktäschel, P. Lewis, A. Bakhtin, Y. Wu, A. H. Miller, and S. Riedel, "Language models as knowledge bases?" *arXiv preprint arXiv:1909.01066*, 2019.

[167] C. Lawrence, B. Kotnis, and M. Niepert, "Attending to future tokens for bidirectional sequence generation," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China: Association for Computational Linguistics, 2019, pp. 1–10.

[168] A. Fan, M. Lewis, and Y. Dauphin, "Hierarchical Neural Story Generation," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, 2018, pp. 889–898. arXiv: 1805.04833.

[169] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli, "Fairseq: A fast, extensible toolkit for sequence modeling," in *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

[170] J. S. Olson and W. A. Kellogg, *Ways of Knowing in HCI*. Springer Publishing Company, Incorporated, 2014, ISBN: 1493903772.

[171] C. Purdy, X. Wang, L. He, and M. Riedl, "Predicting Generated Story Quality with Quantitative Measures," in *In Proceedings of 14th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE'18)*, 2018.

[172] P. Tambwekar, M. Dhuliawala, A. Mehta, L. J. Martin, B. Harrison, and M. O. Riedl, "Controllable Neural Story Plot Generation via Reward Shaping," *arXiv:1809.10736*, 2019. arXiv: 1809.10736.

[173] N. Justesen, R. R. Torrado, P. Bontrager, A. Khalifa, J. Togelius, and S. Risi, "Illuminating generalization in deep reinforcement learning through procedural level generation," *arXiv preprint arXiv:1806.10729*, 2018.

[174] K. Cobbe, C. Hesse, J. Hilton, and J. Schulman, "Leveraging procedural generation to benchmark reinforcement learning," in *International conference on machine learning*, PMLR, 2020, pp. 2048–2056.

[175] L. Wu, A. Fisch, S. Chopra, K. Adams, A. Bordes, and J. Weston, "Starspace: Embed all the things!" In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

[176] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Text Summarization Branches Out*, Barcelona, Spain: Association for Computational Linguistics, Jul. 2004, pp. 74–81.

[177] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[178] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "SQuAD: 100,000+ questions for machine comprehension of text," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Austin, Texas: Association for Computational Linguistics, 2016, pp. 2383–2392.

[179] M. Shridhar, J. Thomason, D. Gordon, Y. Bisk, W. Han, R. Mottaghi, L. Zettlemoyer, and D. Fox, "ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[180] O. Levy, M. Seo, E. Choi, and L. Zettlemoyer, "Zero-shot relation extraction via reading comprehension," in *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, Vancouver, Canada: Association for Computational Linguistics, 2017, pp. 333–342.

[181] S. H. M. Mehr, M. Craven, A. I. Leonov, G. Keenan, and L. Cronin, "A universal system for digitization and automatic execution of the chemical synthesis literature," *Science*, vol. 370, no. 6512, pp. 101–108, 2020. eprint: https://science.sciencemag.org/content/370/6512/101.full.pdf.

[182] T. Gebru, J. Morgenstern, B. Vecchione, J. W. Vaughan, H. Wallach, H. Daumé III, and K. Crawford, "Datasheets for datasets," *arXiv preprint arXiv:1803.09010*, 2018.

[183] S. Wiseman, S. Shieber, and A. Rush, "Challenges in data-to-document generation," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, Copenhagen, Denmark: Association for Computational Linguistics, 2017, pp. 2253–2263.

[184] L. Martin, P. Ammanabrolu, X. Wang, W. Hancock, S. Singh, B. Harrison, and M. Riedl, "Event representations for automated story generation with deep neural nets," in *AAAI Conference on Artificial Intelligence*, 2018.

[185] N. Walton *et al.*, *AI Dungeon*, https://play.aidungeon.io/, 2020.

VITA

Prithviraj Venkata Ammanabrolu, also called either Prithvi or Raj by those trying not to get tongue-tied (and depending on what country you're in), was a student at Georgia Tech—living in Atlanta, Georgia—for the last six years prior to writing this. He was first an undergrad and then a PhD student, both in Computer Science. During this period, he spent time away from school on multiple internships that resulted in some of the work shown in this dissertation.

Before all of that, he did his high school in Bangalore, India. Things even before that get fuzzier, as he has never really spent more than a year in a place prior to high school and did not really attend a school in a standard fashion. He has lived across Asia—places like Hong Kong, Singapore, and various parts of East Asia—and has traversed the breadth of North America. All of this started with his hometown of Visakhapatnam, Andhra Pradesh, India—which he still considers to be his only real home.

In another life, he was a martial artist—a first dan black belt in Tae Kwon Do and later a mixed martial artist. His other hobbies, during the brief moments such things are allowed, involve immersing himself in high fantasy and sci-fi novels or sequestering himself in the middle of a forest somewhere. (Video/Text) Games do not count as hobbies because they are part of his research, or so he tells himself anyway.