Playing Text-Adventure Games with Graph-Based Deep Reinforcement Learning

Paper by Prithviraj Ammanabrolu and Mark O. Riedl Georgia Institute of Technology

Atlanta, GA

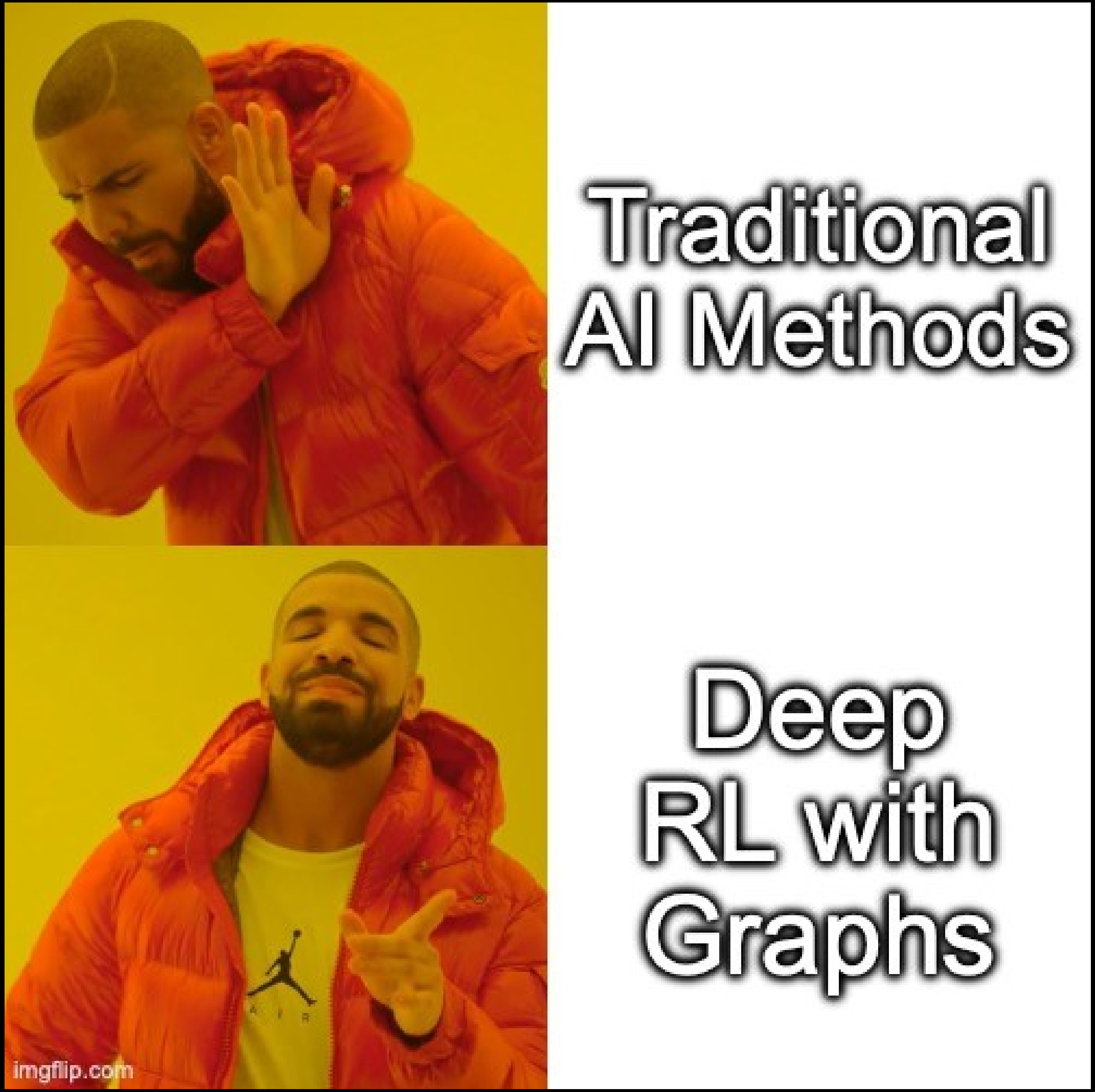
Presentation by Lalith Avinash Donkina

Introduction

Text-based adventure games present a complex environment where the agent only has partial observability of the world.

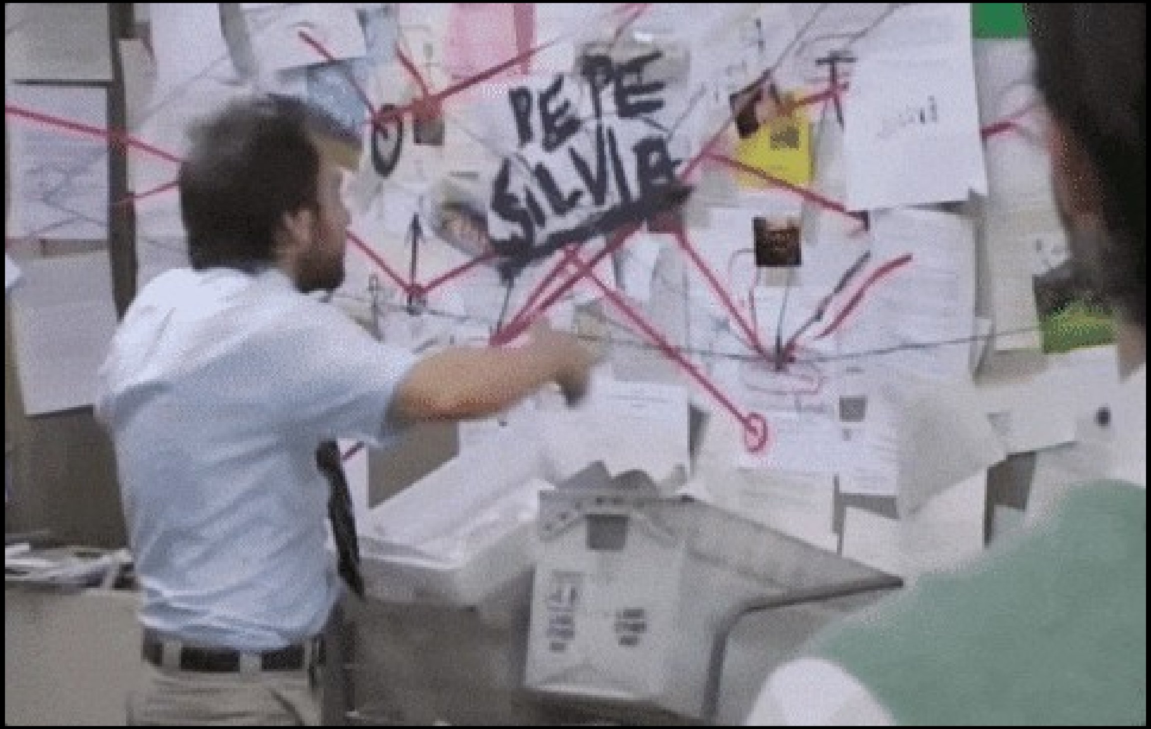
Actions and states in these games are represented through natural language, making the action space combinatorically large.

The goal: Develop an RL-based architecture that helps the AI efficiently explore and make decisions in text-based games.



Understanding the POMDP Structure in Text Games

- Text games can be modeled as Partially Observable Markov Decision Processes (POMDPs).
- A POMDP has seven key components: 〈S, T, A, Ω, O, R, γ〉 representing states, transitions, actions, observations, and rewards.
- The agent must infer the current state and select actions from text-based clues, making it a unique challenge for RL.

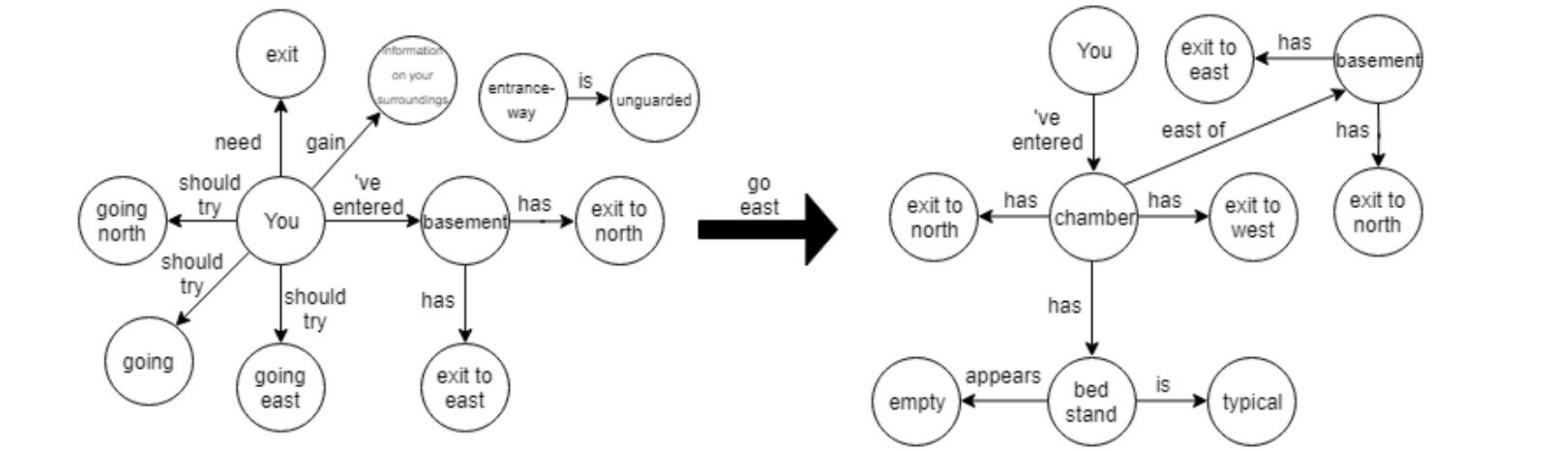


AI Agent’s struggle to understand game states through incomplete information.

Building a Persistent Memory with Knowledge Graphs

- Knowledge graphs represent relationships between entities, allowing the agent to ‘remember’ connections in the game world.
- In this setup, the graph consists of RDF triples: 〈subject, relation, object〉, such as 〈chamber, has, bed stand〉.
- This persistent memory lets the AI prune irrelevant actions, focusing only on meaningful choices in the game.

The knowledge graph helps the Agent dodge irrelevant actions.



You've entered a basement. You try to gain information on your surroundings by using a technique you call "looking." You need an unguarded exit? You should try going east. You don't like doors? Why not try going north, that entranceway is unguarded.

You've entered a chamber. You can see a bed stand. The bed stand is typical. The bed stand appears to be empty. There is an exit to the north. Don't worry, it is unblocked. There is an unblocked exit to the west.

Figure 1: Graph state update example given two observations

Reducing the Action Space with Knowledge Graphs

- Games have a massive action space.
- Action pruning via the knowledge graph restricts this space, scoring actions to keep only the most relevant choices.
- Scoring criteria: +1 for each object in the action present in the graph, +1 for a valid path between objects in the graph.

KG-DQN focuses only on important actions to avoid wasting time.

KG-DQN Architecture

- KG-DQN leverages Q-learning, estimating Q-values for each state-action pair, enabling the agent to select actions with the highest expected reward.
- Graph attention mechanisms help the model focus on relevant nodes in the knowledge graph, refining the action space.
- The architecture includes modules for embedding the graph and computing Q-values with LSTM-encoded observations.

KG-DQN selecting the highest Q-value action for maximum reward.

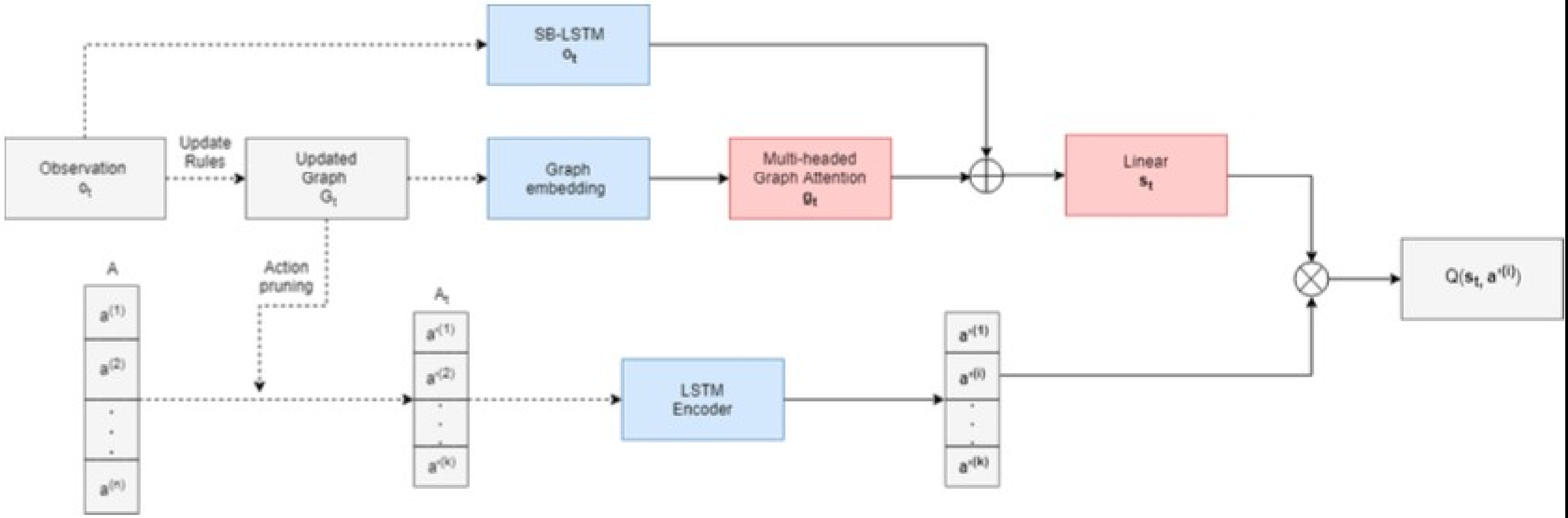


Figure 2: KG-DQN architecture, blue shading indicates components that can be pre-trained and red indicates no
pre-training. The solid lines indicate gradient flow for learnable components.

Training with Reinforcement Learning and Pre-Training

- The model uses Q-learning with experience replay and prioritized sampling.
- Pre-training uses a Question-Answering (QA) paradigm, where the agent learns from a supervised approach to ask itself “What is the right action?”
- QA-based pre-training serves as transfer learning, allowing the agent to apply learned behaviors to new game contexts.

KG-DQN strengthening through experience replay and QA pre-training.

Experimental Results and Evaluation - Evaluating KG-DQN

- Experiments were conducted in the TextWorld framework with small and large game settings.
- Results: KG-DQN converged 40% faster on small games and 25% faster on large games than baselines like LSTM-DQN.
- KG-DQN achieved better performance by completing quests with fewer steps, demonstrating its efficiency and improved decision-making.

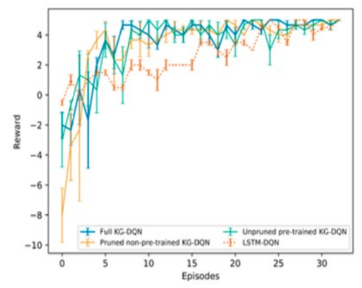


Figure 3: Reward learning curve for select experiments with the small games.

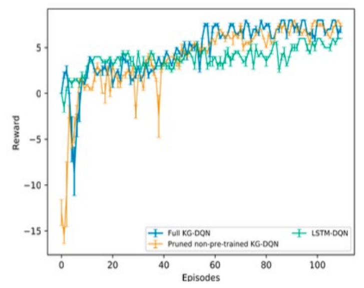


Figure 4: Reward learning curve for select experiments with the large games.

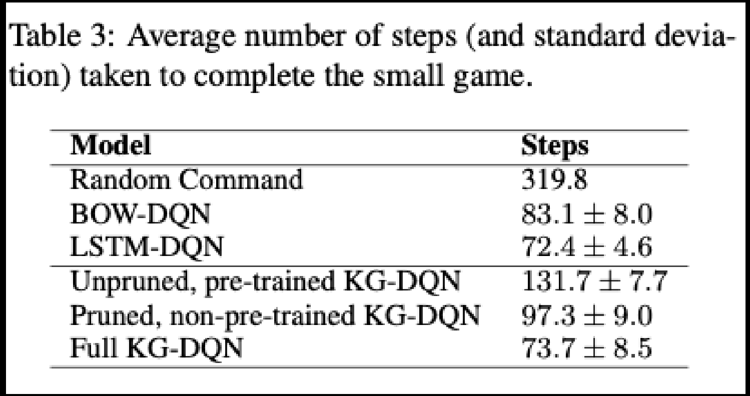


Table 3: Average number of steps (and standard deviation) taken to complete the small game.

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

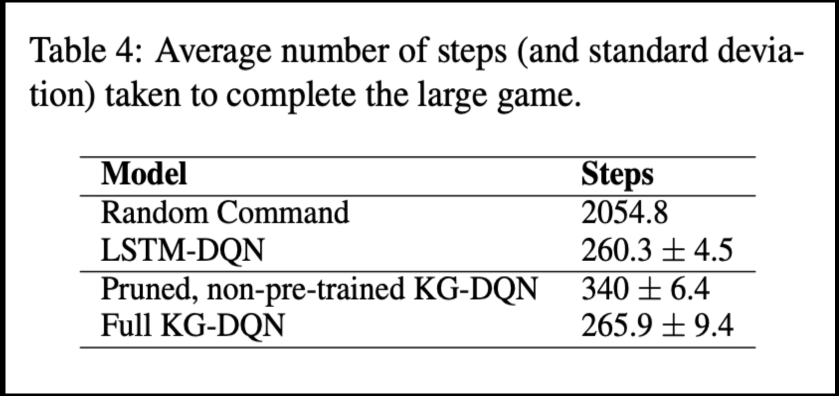
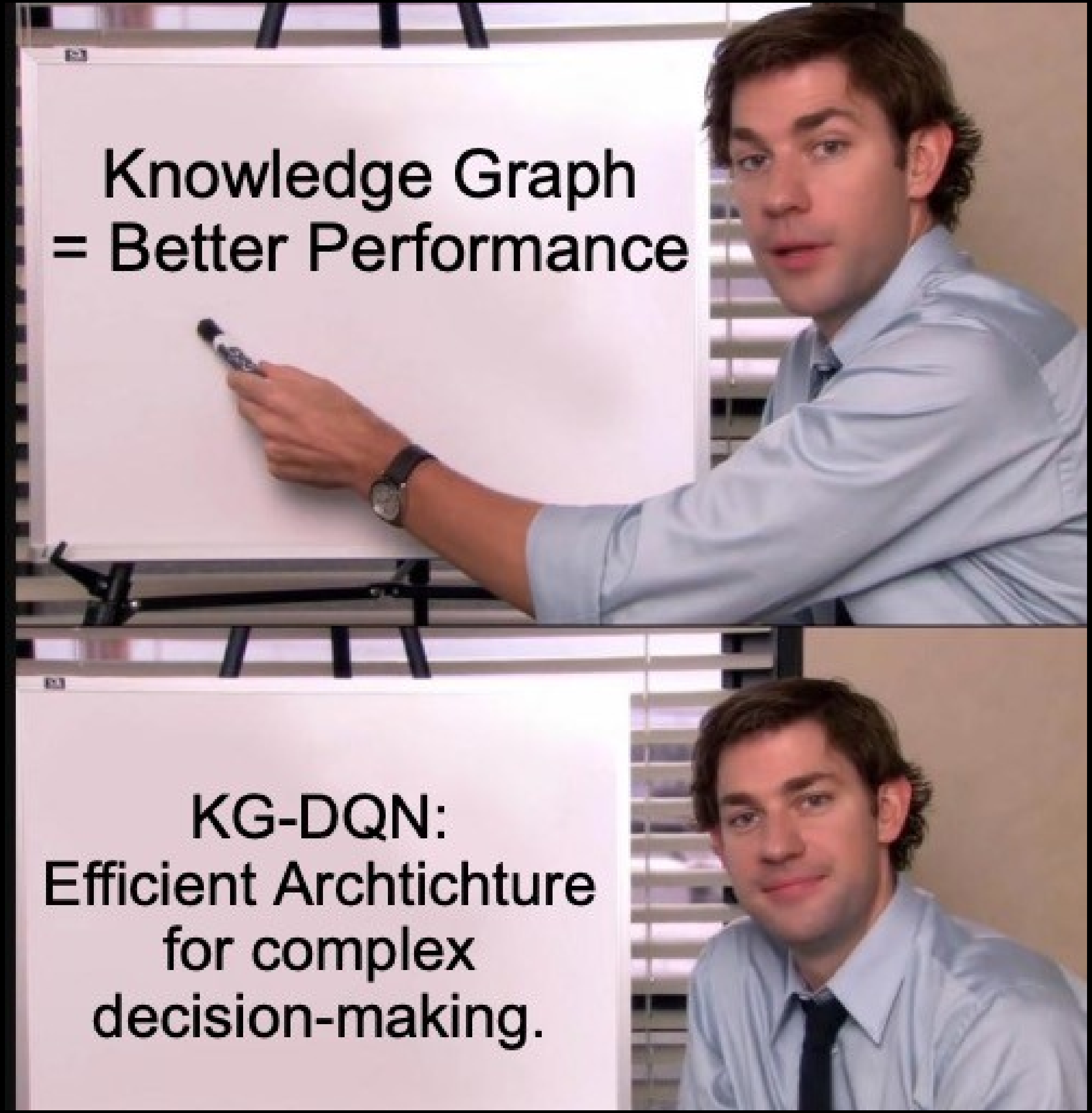


Table 4: Average number of steps (and standard deviation) taken to complete the large game.

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

Conclusion and Future Work

- KG-DQN leverages knowledge graphs to enhance memory, focus, and efficiency in text-based games.
- Future work could extend this approach to more complex environments and explore advanced pre-training.
- This method showcases KG_DQN potential in IF games, bringing us closer to real-world applications of RL in environments with partial observability.



Thank You!