Solving Text-Based Games with Large Language Models

Saeed Hedayatian
Alejandro Murillo-González

Text-Based Games

Problem Setting

- Textual **observations** and **actions**
- Game engine provides a list of **valid actions** at each state
- Game provides a **scalar reward** after each action (for RL)
- Goal: complete the game, maximize total reward

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**

Current Approaches

Reinforcement Learning

- Large, dynamic action space
- Millions of interactions with the environment
- Game-specific

\+ SOTA on almost all games!

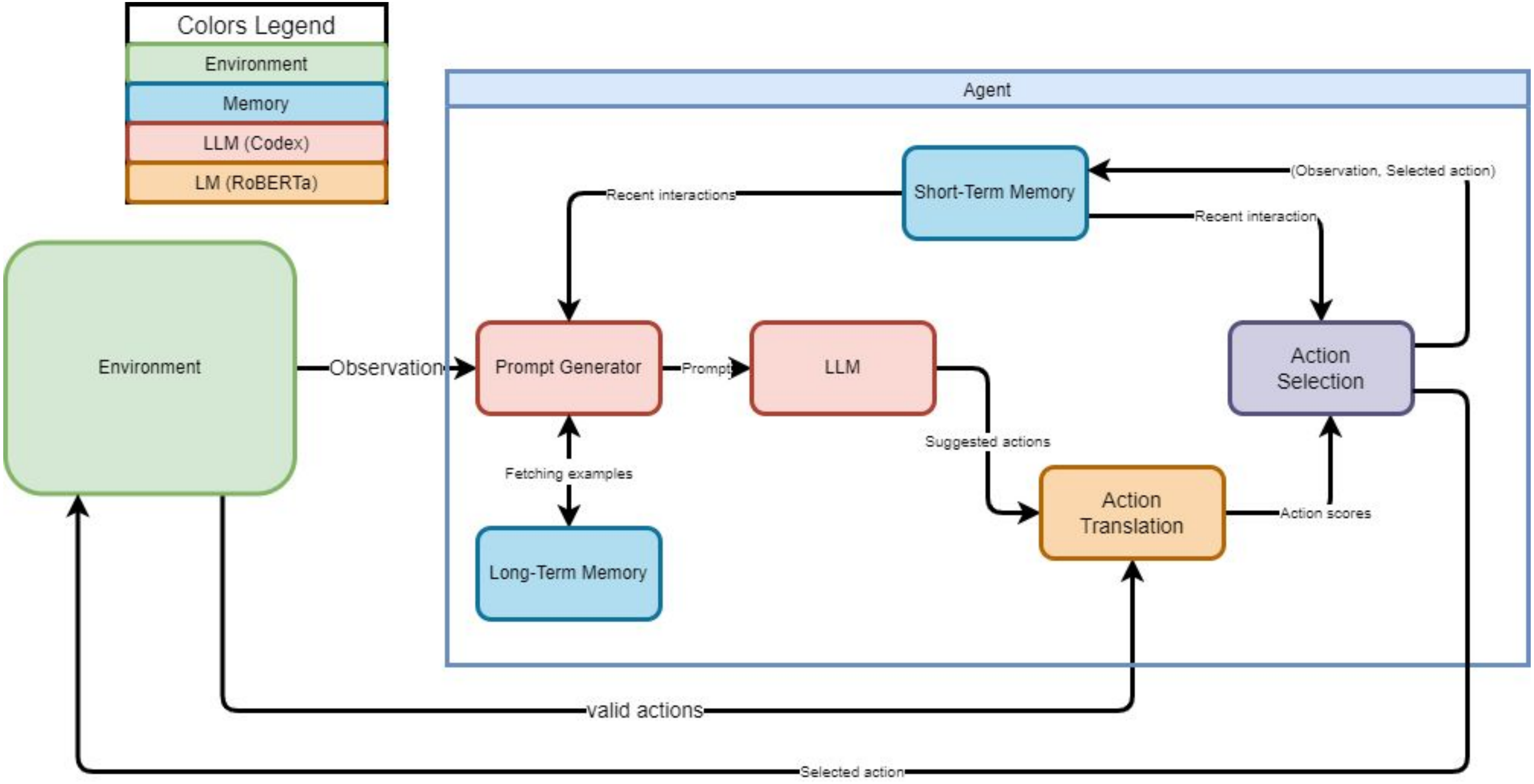
Heuristic and Template Matching

- Hand-crafted rules
- Average performance

\+ No training!

\+ General agent

LLM Agent


Colors Legend
Environment
Memory
LLM (Codex)
LM (RoBERTa)
Agent
Short-Term Memory
(Observation, Selected action)
Recent interactions
Recent interaction
Environment
Observation
Prompt Generator
Prompt
LLM
Action Selection
Suggested actions
Fetching examples
Action Translation
Action scores
Long-Term Memory
valid actions
Selected action

Few-Shot Setting

- Once every n (= 8) episodes, add the collected episodes to long memory:
  - For each state in an episode, compute its reward-to-go
  - Based on the future rewards, label the actions with Very Good, Good, Bad, Very Bad
- When creating the prompt for the current observation:
  - Fetch similar experiences from long-term memory
  - Select k (=3) with largest future reward (positive or negative)
- Report mean, std. of final game score for each n-episode cycle
- In the last cycle, action-selection temperature is very low → Agent becomes more deterministic (But there is still stochasticity)

Prompt Structure

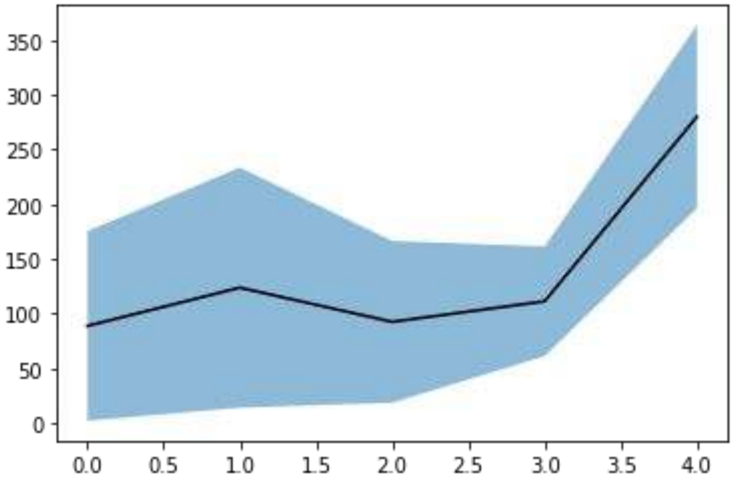
1. Long-term memory examples
2. Instructions
3. Short-term memory (current episode's context)

<START>
<STATE> …
<[VERY][GOOD] ACTION> …
.
.
.
<END>

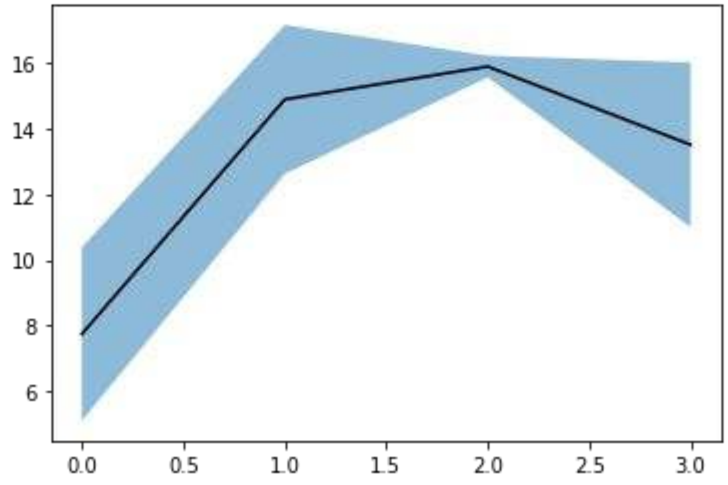
<INSTRUCTION> Repeat GOOD actions
<INSTRUCTION> Avoid BAD action
<INSTRUCTION> Do NOT die
.
.
.

<START>
<STATE> …
<ACTION> …
.
.
.
<STATE> …
<ACTION> [LLM will suggest continuation]

Results



Detective



Library

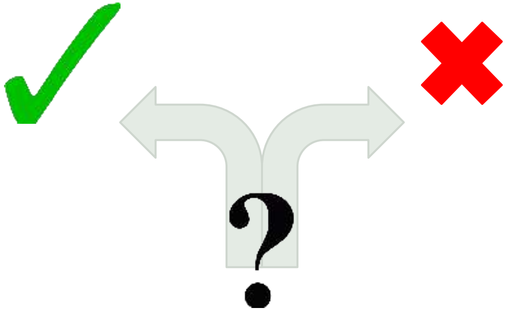
Horizontal axis: # of episodes in long-term memory
Vertical axis: Final score

Results

Game (Max Score)	LLM Agent	KG-A2C	CALM (GPT-2)	NAIL	Q*BERT
Detective (360)	**280**	207.9	**289.7**	136.9	246.1
Library (30)	**13.5**	**14.3**	9	0.9	10
Zork 1 (350)	18	**34**	30.4	10.3	**33.6**
Zork 3 (7)	**2**	0	0.5	**1.8**	-
Deephome (300)	**9**	1	1	**13.3**	1

Takeaways

+ Very good few-shot performance
+ Better than RL at exploration and sparse-reward settings
- Doesn’t avoid Bad actions!
- performance varies across runs due to LLM sampling stochasticity → If we get unlucky in the beginning, it will be hard to recover
- List of valid actions is not always complete



Knowledge Graph Agent

Related Work

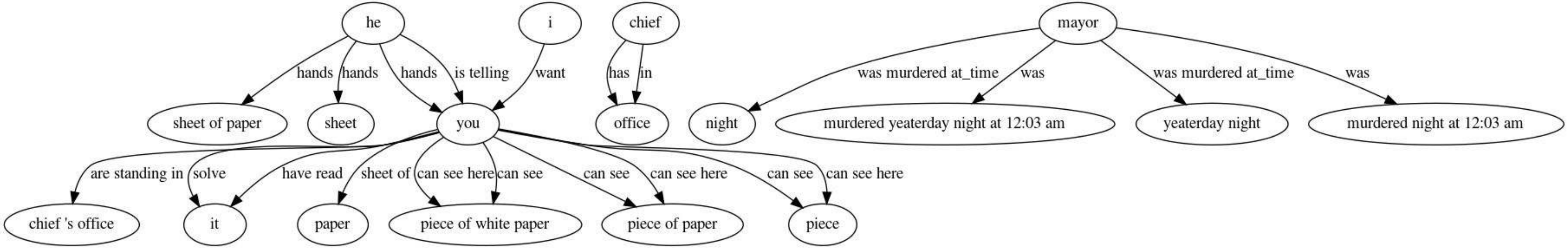
- “The dual uses of the knowledge graph to reason about game state and to constrain natural language generation are the keys to **scalable exploration** of combinatorially large natural language actions” [1]
- In [2], “incorporating a knowledge graph into a reinforcement learning agent results in convergence to the highest reward more than **40% faster than the best baseline**”.
- In [3], **commonsense knowledge is encoded** in the model using a graph, aiming to facilitate story context and generating coherent predictions for the task of story ending generation.
- In [4], a weighted task-related graph is learned from a large scale knowledge base, which forms a Markov chain that is then used to **probabilistically walk along it** and generate instructions.

Graph-Generation

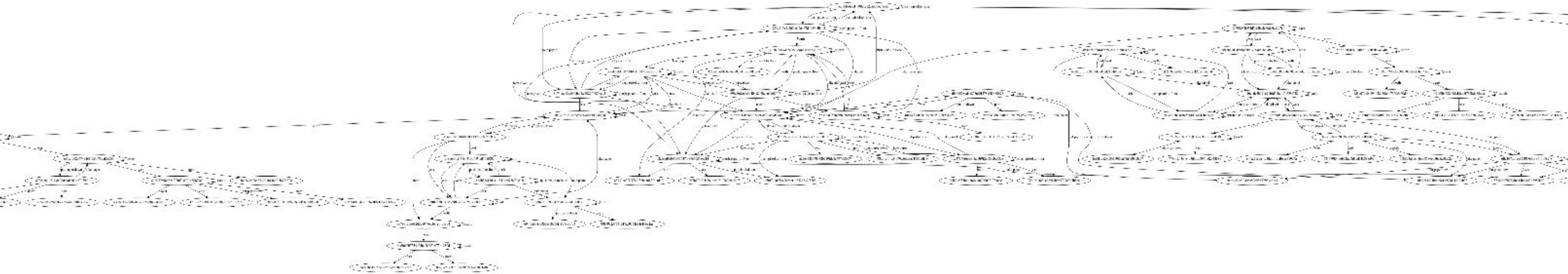
We explore the game until it is possible to build a *depth n* graph, whose paths can then be used to ask the LLM which one to pick.

- At each stage, the graph is built using the current scene description and the agent’s available objects.
- The relationships are extracted using Stanford’s OpenIE [5]
    - The relation schema does not need to be specified in advance.
    - Extracts self-contained clauses from long sentences.

Observation for one scene in “Detective”



Connections between 2.000 possible states in “Detective”



Results & Challenges

- The LLM tends to lead the agent towards infinite loops.
    - This can be solved constraining the knowledge graph (state hashes, etc)
- The LLM doesn’t seem to understand the effect of transitions between the states.
- Performance improves when sampling possible next actions using CALM [6], instead of the game-engine’s set of suggested actions.
- Most works leveraging knowledge graphs do so with the help of RL agents.

References

[1] Ammanabrolu, P., & Hausknecht, M. (2020). Graph constrained reinforcement learning for natural language action spaces. *arXiv preprint arXiv:2001.08837*.

[2] Ammanabrolu, P., & Riedl, M. O. (2018). Playing text-adventure games with graph-based deep reinforcement learning. *arXiv preprint arXiv:1812.01628*.

[3] Guan, J., Wang, Y., & Huang, M. (2019, July). Story ending generation with incremental encoding and commonsense knowledge. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 33, No. 01, pp. 6473-6480).

[4] Ammanabrolu, P., Broniec, W., Mueller, A., Paul, J., & Riedl, M. O. (2019). Toward automated quest generation in text-adventure games. *arXiv preprint arXiv:1909.06283*.

[5] Angeli, G., Premkumar, M. J. J., & Manning, C. D. (2015, July). 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)* (pp. 344-354).

[6] Yao, S., Rao, R., Hausknecht, M., & Narasimhan, K. (2020). Keep CALM and explore: Language models for action generation in text-based games. *arXiv preprint arXiv:2010.02903*.

Thank You!