Using LLM for Reward Shaping in Sparse RL Environments

Your Name
`Reward-Seekers`

1 Motivation

Sparse-reward environments are notoriously difficult for reinforcement learning agents due to their delayed and infrequent feedback. Reward shaping can dramatically accelerate learning, but it is tedious and error-prone when done manually. This project investigates whether LLM can be used as a semi-automated assistant for designing reward functions, ultimately aiming to reduce human effort while maintaining or improving performance.

2 Related Topics

This project relates to several key ideas from the course: (1) Exploration in Reinforcement Learning , (2) Actors and Critics (3) reproducibility and empirical experimentation. It also draws inspiration from the EUREKA paper [1], which explores large-scale LLM-generated rewards, although we take a smaller and manual prompt-based approach.

3 Idea

The goal is to use LLM as a design assistant to shape rewards in the MiniGrid-DoorKey-5x5-v0 environment. We will compare three variants: (1) baseline sparse reward, (2) manually shaped reward, and (3) reward shaped via LLM prompts. We hypothesize that LLM-shaped reward functions will outperform both manual and sparse ones in terms of learning efficiency. The LLM will not generate rewards at runtime but will suggest reward logic based on training trajectories and task descriptions.

Algorithm 1 Simplified reward shaping iteration.

Require: MiniGrid environment e, PPO agent A, LLM model
  for $i = 1$ to N **do**
    Run PPO on e with current reward r_i
    Collect logs of agent behavior
    Prompt LLM with behavior + task goals
    Generate new reward function r_{i+1}
  end forreturn final policy π

4 Experiments

Environments & Metrics We will use MiniGrid-DoorKey-5x5-v0, which presents a sparse-reward navigation task. Metrics include mean episodic return, steps to convergence, and learning curve slope. Optionally, success rate and number of door/key interactions may be tracked.

Experimental Scope We will compare:

- Sparse baseline (default environment reward)
- Manual shaping (for example, +0.1 for picking key, +0.1 for opening door)
- LLM-generated shaping (from iteratively prompted reward code)

Each condition will be tested with 10 random seeds. No hyperparameter tuning is planned beyond standard PPO. One ablation may involve removing the key from LLM-shaped reward.

5 Timeline

- **Research:** 1–2 days — background reading, prompt testing
- **Implementation:** 2–4 days — wrappers, logging, prompt interface
- **Experiments:** 2–3 days — run
- **Analysis:** 1 day — plot and compare results, LL; behavior
- **Reporting:** 3 days — poster + writeup in Overleaf

References

[1] Steven Wang, Amy Zeng, Christian Tjandraatmadja, Shashank Srivastava, Yecheng Jason Ma, Yuhuai Wu, and John Schulman. Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv:2310.00988*, 2023.