LLM-Generated Reward Shaping at Student Scale

Leon Biermann

1 Motivation

Text2Reward showed that a single LLM-written reward script can replace painstaking manual shaping, reaching $> 94\%$ success on 23 continuous-control robotics tasks. But two practical gaps matter for everyday RL users:

1. **Model variability.** Is a cutting-edge "thinking" model like Gemini 2.5 Pro uniquely good, or can a lighter-weight variant such as GPT-4 mini generate rewards that work nearly as well?
2. **Domain breadth.** Does language-driven shaping help on lightweight, *discrete* benchmarks such as CartPole or MiniGrid that dominate introductory courses and quick research prototypes?

Answering both—on a single-GPU laptop budget—will clarify whether "prompt-to-reward" is a robust everyday tool or a niche trick that requires top-tier models and heavy simulators.

2 Research Question

Confirmatory–comparative study (student-scale).

- **RQ1 (Model Effect).** Between Gemini 2.5 Pro and GPT-4 mini, how much does the choice of LLM affect reward-shaping quality?
- **RQ2 (Domain Generalisation).** Does an LLM-generated dense reward improve learning on a simple classic-control task *and* on a sparse grid-world puzzle versus standard baselines?

3 Related Topics

4 Idea

We adopt the Text2Reward pipeline with two key reductions:

1. **LLM back-ends.** Gemini 2.5 Pro and GPT-4 mini. One *zero-shot* prompt, temperature 0.0, generated offline once per task (with a single retry if code fails).
2. **Target environments.**
   (a) **CartPole-v1** – tiny, well-understood; tests whether shaping is still useful.
   (b) **MiniGrid-LavaGap-S7-v0** – sparse-reward puzzle requiring exploration.
3. **RL algorithms.** PPO for CartPole (0.5 M steps) and DQN for MiniGrid (1 M steps) via Stable-Baselines3.
4. **Evaluation.** Compare against the sparse reward and a minimal hand-tuned dense potential.

Project for Reinforcement Learning lecture

Algorithm 1 Training with an LLM-generated reward

Require: environment e, LLM $\mathcal{M}$, RL algorithm A
Prompt $\mathcal{M}$ with task description $\rightarrow$ reward code r_θ
Integrate r_θ into e to obtain shaped environment $\bar{e}$ **return** policy $\pi \leftarrow$ Train A on $\bar{e}$ for N timesteps

$$\pi \in \Pi, \qquad \pi : \mathcal{S} \rightarrow \mathcal{A} \tag{1}$$

5 Experiments

Environments & Metrics

- **CartPole-v1** — success = average return ≥ 195 over 100 episodes.
- **MiniGrid-LavaGap-S7-v0** — success = $\geq 90\%$ completion over 100 episodes.

Measured values: timesteps-to-success, final success rate, reward script lines-of-code (LOC), and per-step reward latency.

Experimental Scope

Factor	Levels
Reward type	sparse, hand-dense, LLM-shaped
LLM back-end	Gemini 2.5 Pro, GPT-4 mini
Prompt style	zero-shot (fixed)
Temperature	0.0 (fixed)
Random seeds	3

2 envs $\times$ 3 rewards $\times$ 2 LLMs $\times$ 3 seeds $=$ 36 training runs.

Estimated Computational Load

- **CartPole-v1:** 0.5 M steps $\approx$ 10 min per run on an RTX 4060 laptop GPU.
- **MiniGrid-LavaGap:** 1 M steps $\approx$ 45 min per run (largely CPU-bound).

Total $\approx$ 32 GPU-hours. Sequential execution fits in $\sim$1.5 days; each environment can run overnight.