A Reinforcement Learning Pipeline for Financial Reasoning (FinQA)

Jinyoon Kim, Scarlett Yu, Donggen Li

Shared Goal & Problem Definition

The Task: Financial Question Answering (FinQA)

- **Input:** Financial reports containing both unstructured text and structured tables.
- **Challenge:** Multi-step reasoning involves retrieving evidence and performing calculations.
- **Research Goal:** Conduct an ablation study to answer: *How do different RL methods affect efficiency, stability, and performance?*.

Unified Framework

- **Shared Dataset:** FinQA (preprocessed into Question-Context-Answer triples).
- **Shared Metrics:** Exact Match (EM), Numeric Accuracy, and Logical Validity.

Project Structure – Two Parallel Tracks

To isolate variables, we split our research into two distinct methodological tracks.

- **Track 1: Deep Contextual RL**
  - **Agent:** Large Language Model (Llama-3.2-3B).
  - **Goal:** Train a neural agent to learn complex financial reasoning patterns using PPO and GRPO.
- **Track 2: Heuristic Bandits**
  - **Agent:** Multi-Armed Bandit (epsilon-Greedy).
  - Goal: Study exploration-exploitation trade-offs using simple statistical learning. (Teammates will detail Track 2 in subsequent slides

Track 1 Challenge – The Generative Blocker

Initial Approach (Generative RL):

- We initially attempted to train Llama-3 to generate full JSON programs character-by-character.

The Critical Failure:

- The model struggled to learn syntax and reasoning simultaneously.
- **Result:** The JSON parse rate stuck at **12%**, providing a sparse reward signal that made RL training impossible.

The Pivot (Discriminative RL):

- We shifted the objective from **Generation** to **Ranking**.
- **Advantage:** This reduces the burden on the 3B model. It no longer needs to *construct* the answer, only *verify* the correct logic.

Methodology – Action Space & Rewards

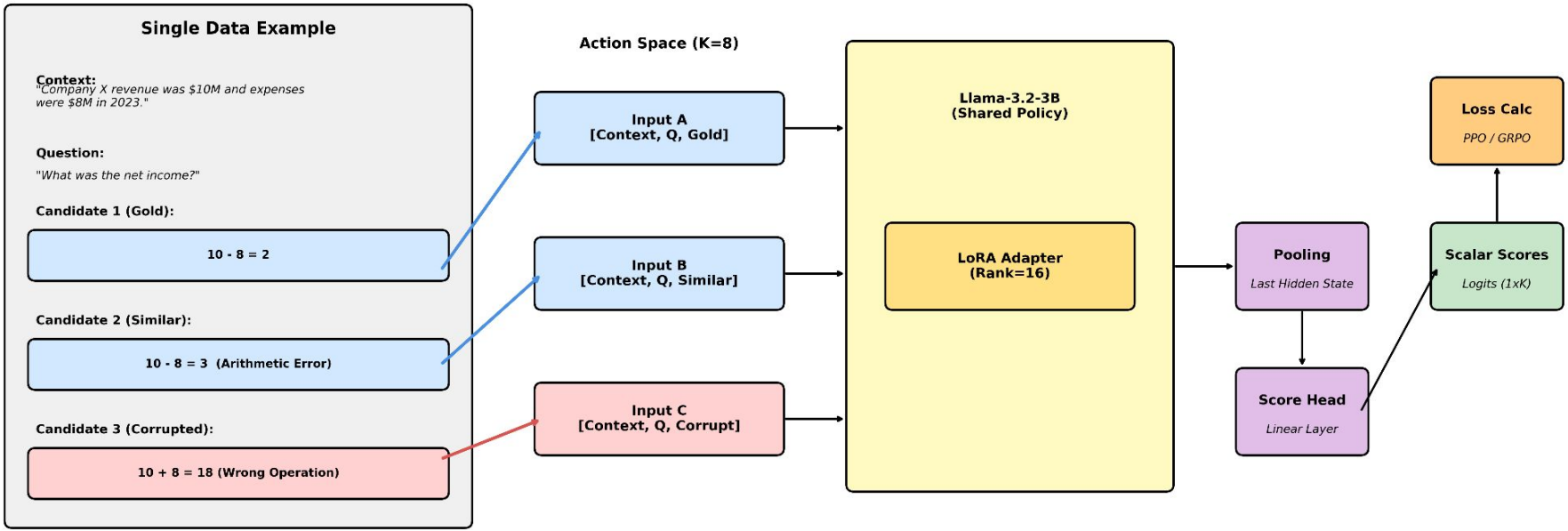
Constructing the Action Space (K=8):

- For every question, we generate a discrete pool of 8 options to force fine-grained discrimination:
    1. **1x Gold:** The ground truth answer.
    2. **Nx Similar:** Plausible distractors (e.g., values perturbed by **± 10%**).
    3. **Mx Corrupted:** Obvious errors (e.g., wrong operations, invalid formats).

Dense Reward Function:

- We replaced sparse binary rewards with a composite signal to guide learning:
- **R_total = 1.0(R_exact) + 0.9(R_numerical) + 0.5(R_logic) + 0.3(R_format)**
- *Note: "Numerical" allows for small rounding errors; "Logic" rewards valid program steps.*

Track 1 System Architecture



Track 1 Algorithms (PPO & GRPO)

To optimize this ranking objective, we implemented and compared two RL algorithms:

- **Algorithm 1: PPO (Proximal Policy Optimization)**
  - **Mechanism:** Optimizes the policy to select the best candidate while staying close to the SFT reference model (using a KL penalty).
  - **Role:** Serves as the stable, standard baseline for Deep RL.
- **Algorithm 2: GRPO (Group Relative Policy Optimization)**
  - **Mechanism:** Samples a group of candidates and optimizes based on their relative advantage compared to the **group mean**.
  - **Hypothesis:** More efficient for ranking tasks as it removes the need for a separate reference model (reducing memory usage).

Experimental Setup

Model Architecture:

- **Base:** Llama-3.2-3B optimized with LoRA (Rank=16).
- **Head:** Linear layer projecting the final hidden state to a scalar reward score.

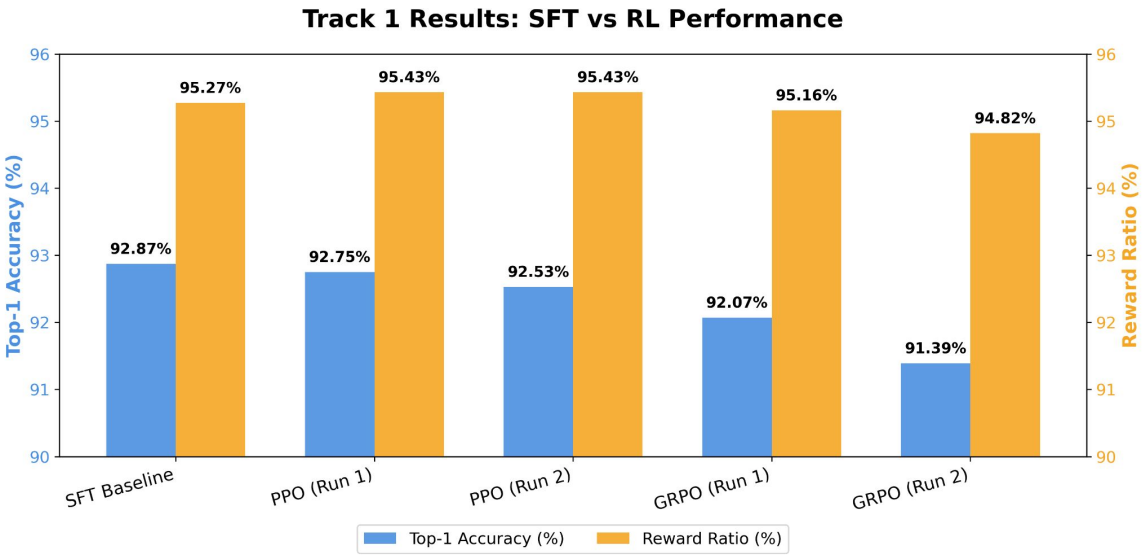
Experimental Configurations:

- **Run 001 (Standard):** 10 Epochs, stronger per-batch optimization (4 PPO epochs).
- **Run 002 (Extended):** 20 Epochs, reduced batch size, designed to test convergence and stability.

Baselines:

- **SFT Baseline:** Trained via Cross-Entropy loss to maximize the probability of the Gold candidate.

Experimental Results



Model Configuration	Top-1 Accuracy	Reward Ratio	Delta from SFT
SFT Baseline	**92.87%**	**95.27%**	**--**
PPO (Run 001)	92.75%	95.43%	-0.12%
PPO (Run 002)	92.53%	95.43%	-0.34%
GRPO (Run 001)	92.07%	95.16%	-0.80%
GRPO (Run 002)	91.39%	94.82%	-1.48%

Observation: The SFT baseline essentially reached the task "ceiling" immediately. PPO remained stable (due to KL penalty), while GRPO degraded significantly in extended training runs.

Analysis – Why Did RL Struggle?

1. The SFT Ceiling Effect:

- The ranking task was structurally simple for the SFT model (92.87% accuracy). There was very little "exploration" room left for RL agents to discover better strategies.

2. Stability vs. Efficiency:

- **PPO:** Remained stable largely due to the **KL-penalty** forcing it to stay close to the reference model.
- **GRPO:** Without a reference model anchor, the policy drifted in Run 002 (-1.48%), likely overfitting to the reward proxy rather than improving reasoning.

3. Overfitting in Extended Runs:

- Doubling the training epochs (Run 002) consistently lowered accuracy, indicating that the agents began "gaming" the reward function or overfitting to noise.

Track1 Conclusion

- **Research Answer:** For discriminative financial reasoning tasks using small LLMs (3B), **Supervised Fine-Tuning is more efficient and stable than Reinforcement Learning.**
- **Efficiency:** SFT required ~50% less compute time (no value function, no rollouts) and achieved the highest accuracy.
- **Stability:** RL methods introduced hyperparameter sensitivity and performance degradation over long training runs.
- **Recommendation:** Discriminative RL is sufficiently solved by SFT in this domain; future research should focus on Generative RL where the action space is infinite.

Extension: Why Perform RL on a Small Model?

- In Track 1, RL brought *limited* gains to the 3B model
- Large models already perform strong after SFT → small room to improve
- Small models (1B) have much weaker reasoning ability → RL effects are amplified

- **Research question:**

  Can RL algorithms like RLOO and DPO significantly improve performance for weaker models?

Experimental Setup

- We run all experiments on the **TinyLlama-1.1B-Chat** model and keep the main FinQA candidate-ranking task unchanged.
- Our study compares three training strategies — **SFT baseline**, **RLOO**, and **DPO** — built entirely on our custom training pipeline.
- To support lightweight CPU-friendly experiments, we implemented new training scripts, a unified evaluation framework, and a run-comparison tool.

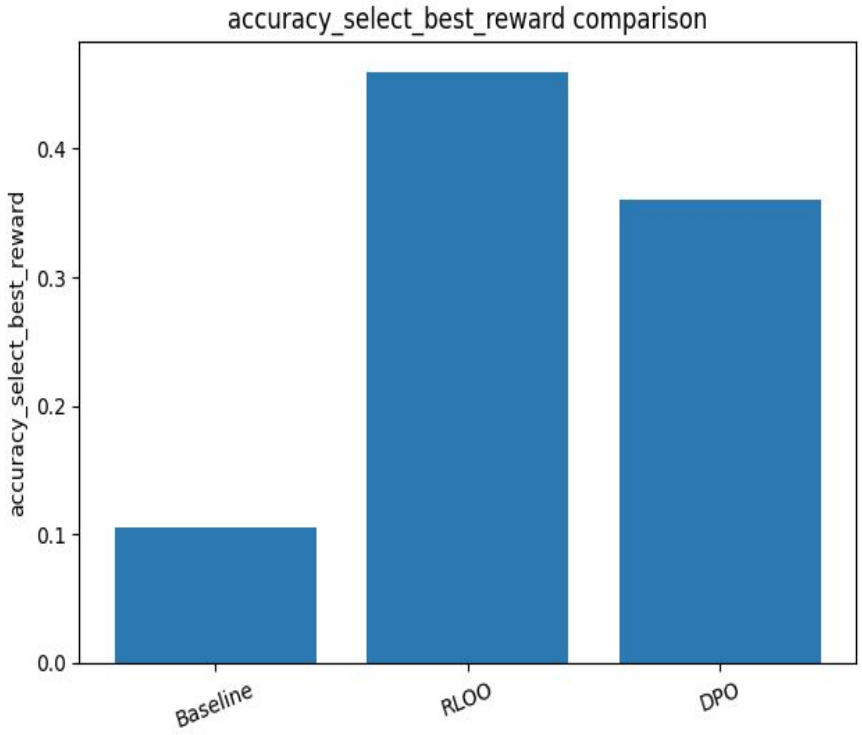
RLOO & DPO — Why These Two Methods?

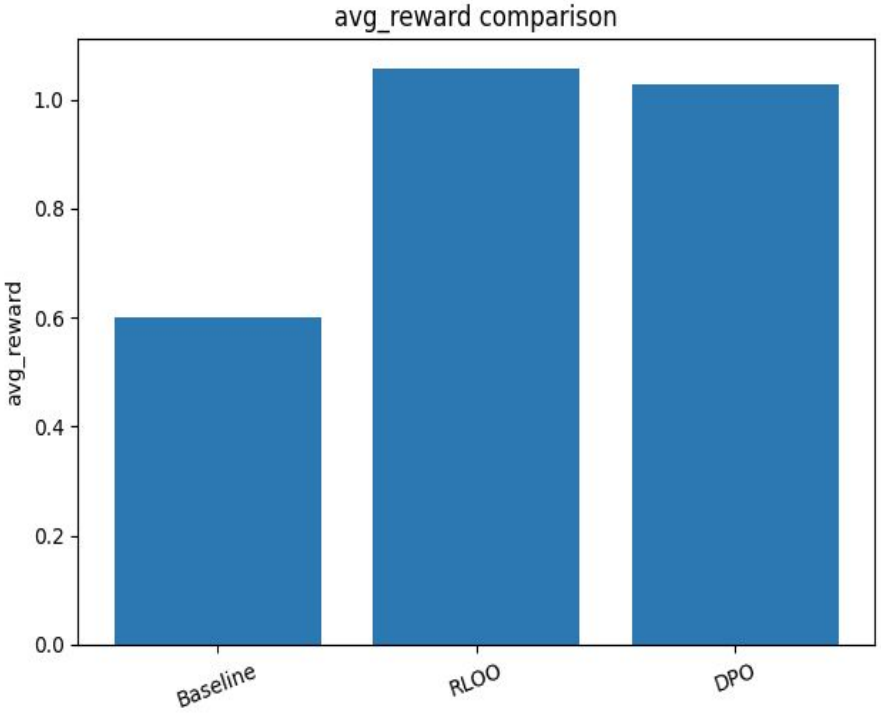
- **RLOO** reduces variance by comparing each action to "leave-one-out" baselines, making it stable for small models with noisy reward signals.

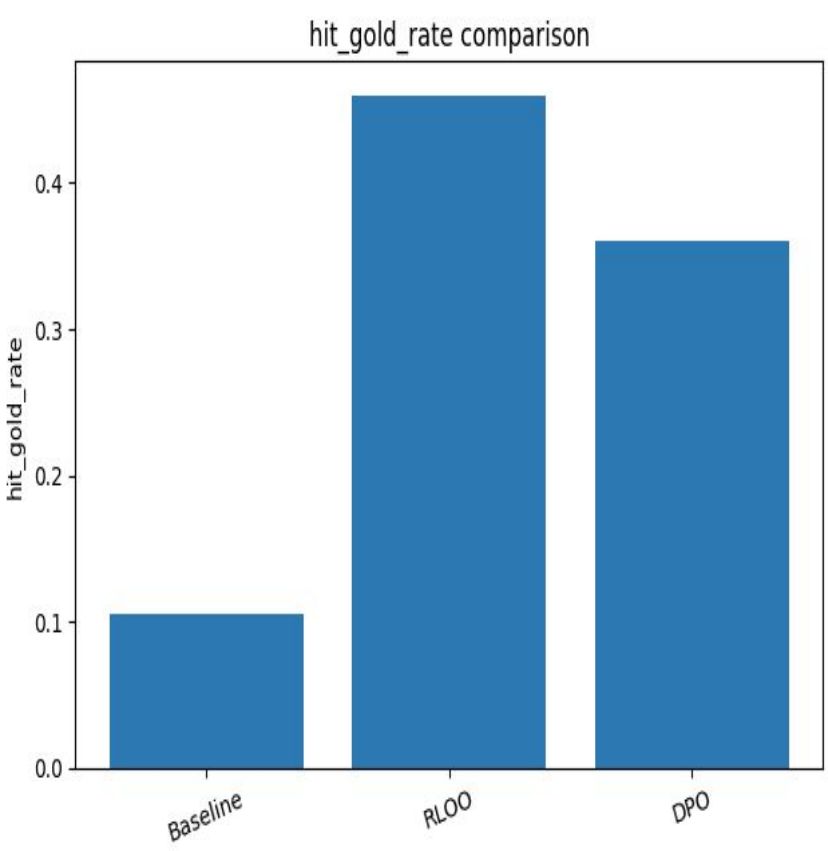
- **DPO** directly optimizes model preferences between good and bad answers, offering a simple and efficient improvement path without policy rollouts.

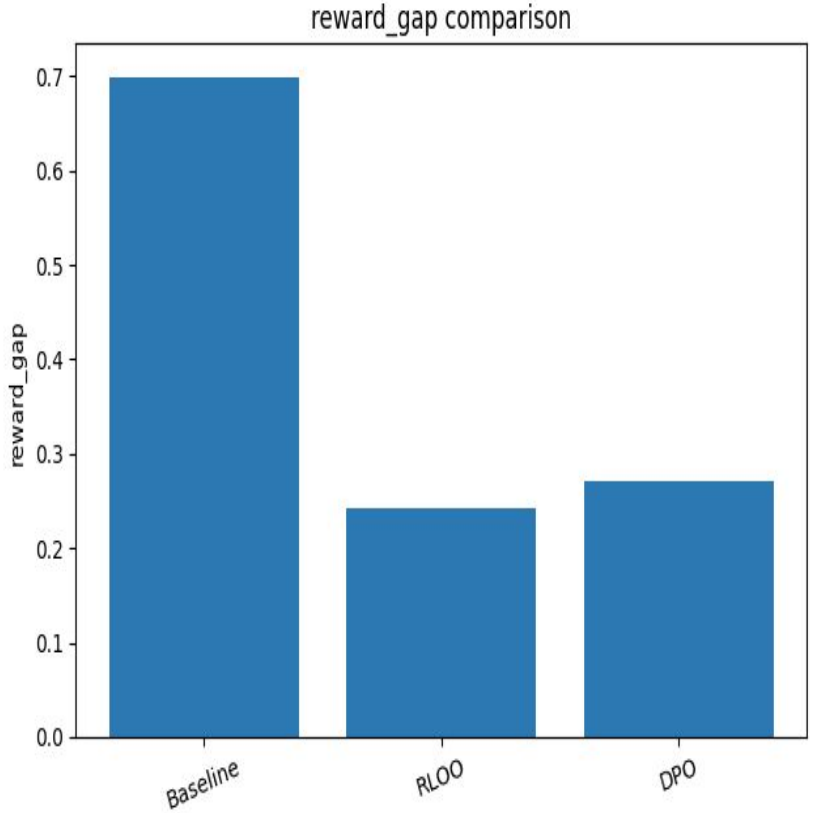
- Both algorithms are lightweight and work well in CPU-only environments, making them ideal for extending RL training to small models.

Performance Comparison (TinyLlama-1.1B)






hit_gold_rate comparison
hit_gold_rate
0.0
0.1
0.2
0.3
0.4
Baseline
RLOO
DPO


reward_gap comparison
reward_gap
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Baseline
RLOO
DPO

Quantitative Results

Method	Accuracy ↑	Avg Reward ↑	Reward Gap ↓	Hit Gold ↑
Baseline	0.105	0.601	0.699	0.105
DPO	0.360	1.028	0.272	0.360
RLOO	**0.460**	**1.057**	**0.243**	**0.460**

- **RL methods significantly outperform the SFT baseline on all metrics.**
- **RLOO shows the strongest gains in selection accuracy (+35%) and reward quality.**
- **Both RLOO and DPO reduce reward gap, indicating better candidate discrimination.**

Conclusion & Takeaways

- **RL can meaningfully improve reasoning performance for small models—much more than for 3B+ models in Track 1.**
- **RLOO is the most effective among lightweight RL methods for small-model FinQA reasoning.**
- **Our extended pipeline (training, evaluation, comparison tools) enables reproducible small-model RL without GPU reliance.**

Track 2 – Bandit-Based Heuristics

Problem Setup

- Task: Select the **relevant table cell** for answering financial questions in **FinQA**.
- Instead of designing one heuristic, treat each heuristic as an **arm in a non-contextual multi-armed bandit**.

Three Heuristic Arms

- **Arm 0 – Token Overlap:**
  Select the cell with the largest lexical overlap with the question.
- **Arm 1 – Maximum Numeric Value:**
  Parse numbers, handle % and parentheses (negative values), choose the **largest numeric entry**.
- **Arm 2 – Uniform Random:**
  Baseline; selects a random table cell.

Track 2 – Bandit-Based Heuristics

Reward Design

- Bandit receives **binary reward {0,1}** based on:
  - Substring match between selected cell and gold evidence strings
  - OR ≥2 shared tokens with an evidence sentence
- This creates a **noisy but meaningful weak reward signal**.

Bandit Algorithm

- **Non-contextual ε-greedy**, ε decays from 0.5 to 0
- Updates sample means **Q** incrementally
- Trained for **10,000 episodes** (each episode is a random FinQA example)

Track 2 Results

Learning Behavior

- Moving-average reward **increases steadily** as ε decays.
- Early phase: mixture of all arms; reward is low due to exploration and random arm.
- Later phase: bandit **commits to the better-performing heuristics**.

Performance Insights

- Arm 2 (random) performs worst → the bandit learns to **avoid** it.
- Arm 0 (token overlap) and Arm 1 (numeric) both yield **meaningfully higher reward**, depending on the dataset distribution.
- Bandit's long-run average reward approaches the reward of the **best static heuristic**, despite noisy rewards.

Conclusion

- Even with noisy supervision and very simple heuristics,
  the bandit can reliably identify the best heuristic over time.
- Demonstrates that bandit-based exploration–exploitation is a viable method for heuristic selection in financial QA.