Implementation Updates

Training Infrastructure Improvements

🎯 Unsloth Optimization for T4 GPUs

- Successfully migrated training from A100 GPUs to T4 GPUs using Unsloth's notebook optimizations
- Enables more cost-effective and accessible training
- Maintains model quality with extended training times

🧱 Response Structure Revisions

- Implemented a structured output format for LLM
- Encourages explicit reasoning rather than only providing decisions
- Reward Functions now evaluate:
    - **Structure Compliance**: Rewards proper tag structure
    - **Reasoning Quality**: Assesses depth, relevance, and logic
    - **Decision Alignment**: Ensures reasoning supports the final decision
- More lenient rewards for faster convergence

Reinforcement Learning Implementation

🤖 Self-Play with PPO

- Implemented **Proximal Policy Optimization (PPO)** for self-play using PyPokerEngine
- Features:
  - 6-player poker games with model instances competing against each other
  - Gameplay data collection for continuous improvement
  - Competitive reinforcement learning for decision refinement

📈 GRPO Training Challenges

- Integrated **Group Relative Policy Optimization (GRPO)** for enhanced learning
- Challenges faced:
    - **Computational Intensity**: Significant time required for updates
    - **Complexity Management**: Poker's vast state space adds difficulty
    - **Long Runtimes**: Full training cycles are resource-intensive
- Further optimization needed for practical training cycles

Enhanced Reward Functions

🛠️ Verbal Descriptions

- **Structure Compliance**:
  Models are rewarded for adhering to predefined tag formats. This ensures consistent output, making evaluation and debugging easier.

- **Reasoning Quality**:
  Reasoning sections are assessed for relevance and logical consistency. More detailed and insightful reasoning results in higher rewards.

- **Decision Alignment**:
  Models earn rewards when the final decision logically follows from the reasoning. Misalignment is penalized to reinforce accurate decision-making.

- **Granular Feedback**:
  Rewards are applied incrementally, reducing punishment for minor errors while

Next Steps

⏩ Training Efficiency

- Explore:
  - Smaller models for faster fine-tuning
  - Algorithmic improvements to reduce compute intensity

⚙️ Reward Function Refinements

- Detect and mitigate reward hacking
- Enhance feedback for better reasoning quality

🧪 Model Evaluation

- Current Method: Track average chip differences in games
- Future Exploration: