Explainable Reinforcement Learning in Board Games

•••

Max Wu
Luke de Vries

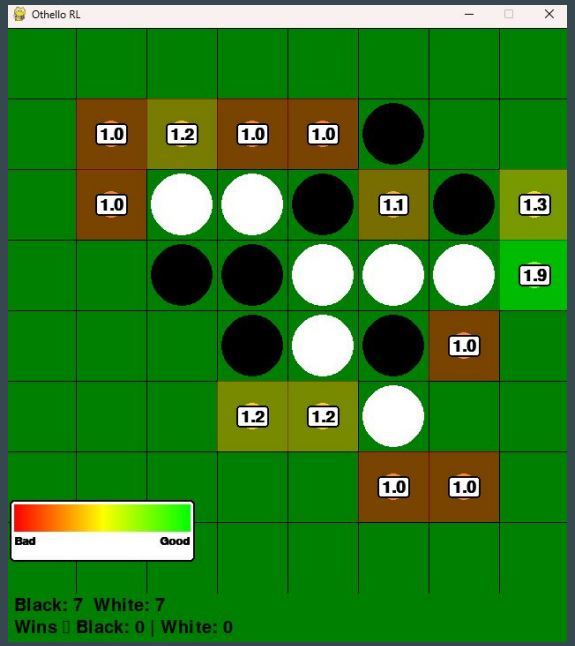
Project Vision

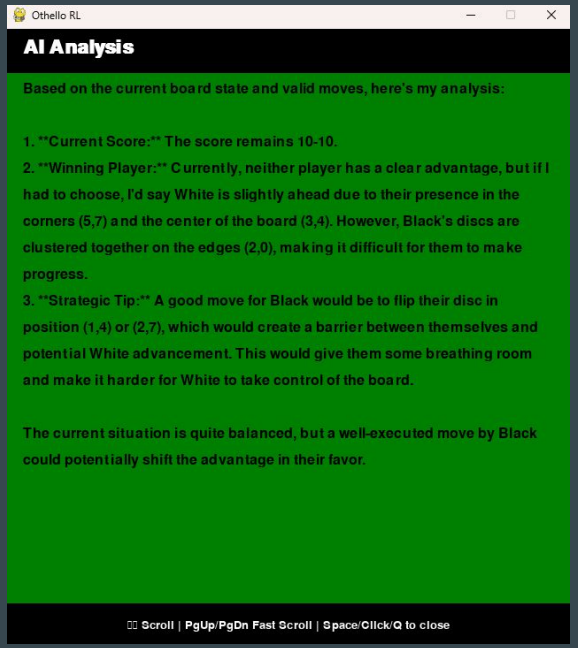
Create a system where AI can:

- Play board games at human-competitive level
- Explain it’s strategic reasoning in natural language
- Provide real-time coaching and feedback for human players
- Make reinforcement learning interpretable and informative in the context of board games

Last Semester

- Implemented Othello with real-time LLM analysis using local llama3
- Color-coded move heatmap based on positional and capture heuristics
- Undo feature with re-analysis so players could learn from mistakes
- Built with Pygame and raw prompt engineering





GameSage

- Increased to 4 games: Chess, Checkers, Go, and Sudoku
- Replaced raw prompting with a DSPy structured pipeline
- Added multi-backend LLM support: Ollama, OpenAI, Anthropic, and Gemini

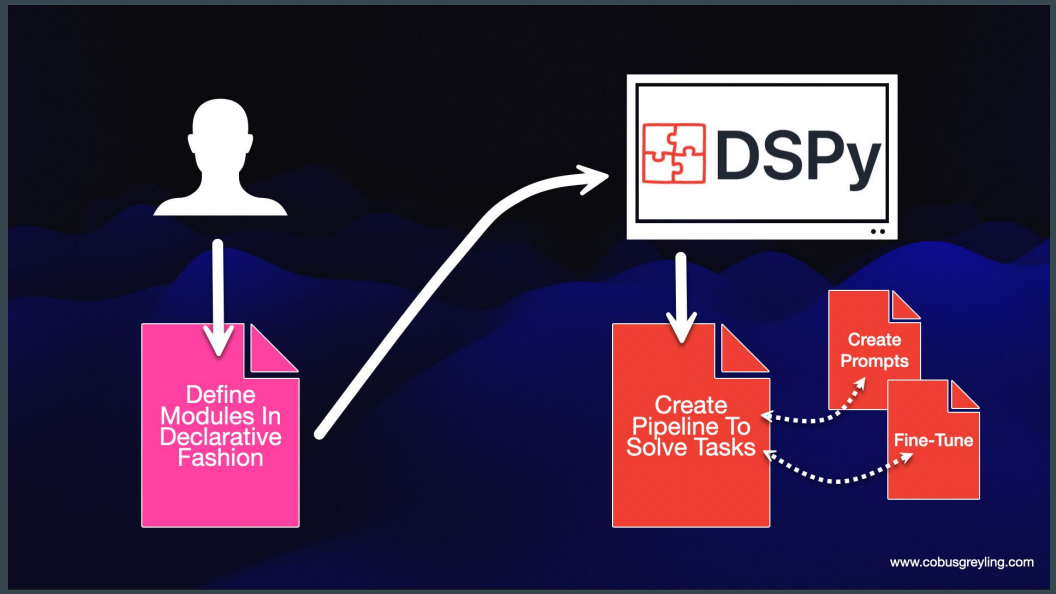
Swapping LLM Backends

Edit `config.py` or set environment variables:

```
# Ollama — local, free, no API key (default)
export GAMESAGE_LLM_BACKEND=ollama
export GAMESAGE_OLLAMA_MODEL=llama3.1      # or qwen2.5, mistral, llama3.3, etc.
export GAMESAGE_OLLAMA_BASE_URL=http://localhost:11434

# OpenAI — GPT-4o recommended
export GAMESAGE_LLM_BACKEND=openai
export GAMESAGE_OPENAI_MODEL=gpt-4o        # or gpt-4o-mini for cheaper runs
export OPENAI_API_KEY=sk-...

# Anthropic — Claude
export GAMESAGE_LLM_BACKEND=anthropic
export GAMESAGE_ANTHROPIC_MODEL=claude-sonnet-4-6   # or claude-haiku-4-5-20251001
export ANTHROPIC_API_KEY=sk-ant-...

# Google Gemini
export GAMESAGE_LLM_BACKEND=gemini
export GAMESAGE_GEMINI_MODEL=gemini/gemini-2.0-flash   # or gemini/gemini-2.5-pro
export GEMINI_API_KEY=AIza...
```

DSPy Pipeline

- GameSageAdvisor uses ChainOfThought on a structured MoveAdvisor signature
- Explicit typed output fields prevent hallucinated terminology
- GameSageCoach combines PositionEvaluator and MoveExplainer
- Illegal-move retry loop with up to 3 attempts before falling back to a random legal move



Architecture

- game engine always validates moves before LLM reads board state
- Research logger automatically records every session to SQLite
- Tracks board states, LLM reasoning, and whether players followed AI advice

```
for attempt in range(LLM_MAX_RETRIES + 1):
    extra = ""
    if illegal_feedback:
        extra = (
            "\n\nNOTE: The following moves are ILLEGAL and must NOT be played: "
            + ", ".join(illegal_feedback)
            + ". Choose a different move from the legal moves list."
        )

    prediction = self.cot(
        game_name=game_name,
        rules_summary=rules_summary,
        board_state=board_state + extra,
        legal_moves=legal_moves_str,
        move_history=history_str,
        player_skill_level=player_skill_level,
    )

    move = prediction.recommended_move.strip()

    # Legality check
    if validate_move_fn is None or validate_move_fn(move):
        return prediction
```

Four Game Modes

- Play: Human vs AI — LLM plays one side and explains each move
- Coach: Human plays both sides; AI comments after every move
- Analyze: Paste any board state for full position analysis
- Puzzle: Step-by-step AI guidance for Chess and Sudoku

Design Norms

- Transparency
- Trust

Related Work

- Ehsan, U. et al. Rationalization: A Neural Machine Translation Approach to Generating Natural Language Explanations. AAAI/ACM Conference on AI, Ethics, and Society, 2018.
- Translates state-action pairs from a game-playing agent into human-like natural language rationales
- Human evaluators rated generated rationales as more satisfying than alternative explanation methods

Rationalization: A Neural Machine Translation Approach to Generating Natural Language Explanations

Upol Ehsan [*†], **Brent Harrison** [*‡], **Larry Chan**[†], and **Mark Riedl**[†]
[†]Georgia Institute of Technology, Atlanta, GA, USA
[‡]University of Kentucky, Lexington, KY, USA
ehsanu@gatech.edu, harrison@cs.uky.edu, larrychan@gatech.edu, riedl@cc.gatech.edu

Abstract

We introduce *AI rationalization*, an approach for generating explanations of autonomous system behavior as if a human had performed the behavior. We describe a rationalization technique that uses neural machine translation to translate internal state-action representations of an autonomous agent into natural language. We evaluate our technique in the Frogger game environment, training an autonomous game playing agent to rationalize its action choices using natural language. A natural language training corpus is collected from human players thinking out loud as they play the game. We motivate the use of rationalization as an approach to explanation generation and show the results of two experiments evaluating the effectiveness of rationalization. Results of these evaluations show that neural machine translation is able to accurately generate rationalizations that describe agent behavior, and that rationalizations are more satisfying to humans than other alternative methods of explanation.

Introduction

Autonomous systems must make complex sequential decisions in the face of uncertainty. *Explainable AI* refers to artificial intelligence and machine learning techniques that can provide human understandable justification for their behavior. With the proliferation of AI in everyday use, explainability is important in situations where human operators work alongside autonomous and semi-autonomous systems because it can help build rapport, confidence, and understanding between the agent and its operator. For instance, a non-expert human collaborating with a robot for a search and rescue mission requires confidence in the robot's action. In the event of failure—or if the agent performs unexpected behaviors—it is natural for the human operator *to want to know why*. Explanations help the human operator understand why an agent failed to achieve a goal or the circumstances whereby the behavior of the agent deviated from the expectations of the human operator. They may then take appropriate remedial action: trying again, providing more training to machine learning algorithms controlling the agent, reporting bugs to the manufacturer, etc.

[*]Harrison and Ehsan equally contributed to this work

Explanation differs from *interpretability*, which is a feature of an algorithm or representation that affords inspection for the purposes of understanding behavior or results. While there has been work done recently on the interpretability of neural networks (Yosinski et al. 2015; Zeiler and Fergus 2014), these studies mainly focus on interpretability for experts on non-sequential problems. Explanation, on the other hand, focuses on sequential problems, is grounded in natural language communication, and is theorized to be more useful for non-AI-experts who need to operate autonomous or semi-autonomous systems.

In this paper we introduce a new approach to explainable AI: *AI rationalization*. AI rationalization is a process of producing an explanation for agent behavior *as if a human had performed the behavior*. AI rationalization is based on the observation that there are times when humans may not have full conscious access to reasons for their behavior and consequently may not give explanations that literally reveal how a decision was made. In these situations, it is more likely that humans create plausible explanations on the spot when pressed. However, we accept human-generated rationalizations as providing some lay insight into the mind of the other.

AI rationalization has a number of potential benefits over other explainability techniques: (1) by communicating like humans, rationalizations are naturally accessible and intuitive to humans, especially non-experts (2) humanlike communication between autonomous systems and human operators may afford human factors advantages such as higher degrees of satisfaction, confidence, rapport, and willingness to use autonomous systems; (3) rationalization is fast, sacrificing absolute accuracy for real-time response, appropriate for real-time human-agent collaboration. Should deeper, more accurate explanations or interpretations be necessary, rationalizations may need to be supplemented by other explanation, interpretation, or visualization techniques.

We propose a technique for AI rationalization that treats the generation of explanations as a problem of *translation* between ad-hoc representations of states and actions in an autonomous system's environment and natural language. To do this, we first collect a corpus of natural language utterances from people performing the learning task. We then use these utterances along with state information to train an encoder-decoder neural network to translate between state-action information and natural language.

GameSage - Applying the Same Principle

- LLM receives only serialized board state and move — internal engine weights never exposed
- MoveAdvisor takes observable state inputs, produces typed rationale outputs

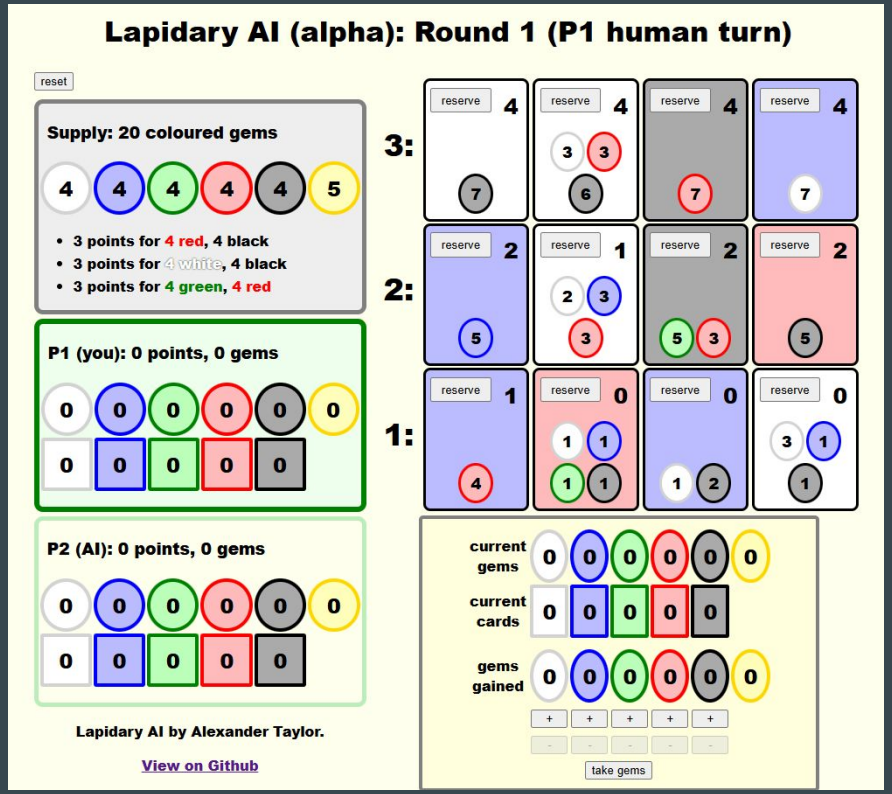
```python
class MoveAdvisor(dspy.Signature):
    """Analyze a board game state and recommend the best move with a full explanation."""

    game_name: str = dspy.InputField(desc="Name of the game being played")
    rules_summary: str = dspy.InputField(desc="Short rules blurb for context")
    board_state: str = dspy.InputField(desc="Board serialized as structured text")
    legal_moves: str = dspy.InputField(desc="Comma-separated list of legal moves")
    move_history: str = dspy.InputField(desc="Recent move history for context")
    player_skill_level: str = dspy.InputField(desc="beginner, intermediate, or advanced")

    strategic_reasoning: str = dspy.OutputField(desc="Internal step-by-step strategic analysis")
    recommended_move: str = dspy.OutputField(desc="The single best move in the game's standard notation")
    explanation: str = dspy.OutputField(desc="Plain English explanation tailored to the player's skill level")
    alternative_moves: str = dspy.OutputField(desc="2-3 alternative moves with brief tradeoff notes")
    key_concepts: str = dspy.OutputField(desc="1-3 strategic concepts this position demonstrates, e.g. 'center control', 'forced capture'")
```

Future Work

Splendor

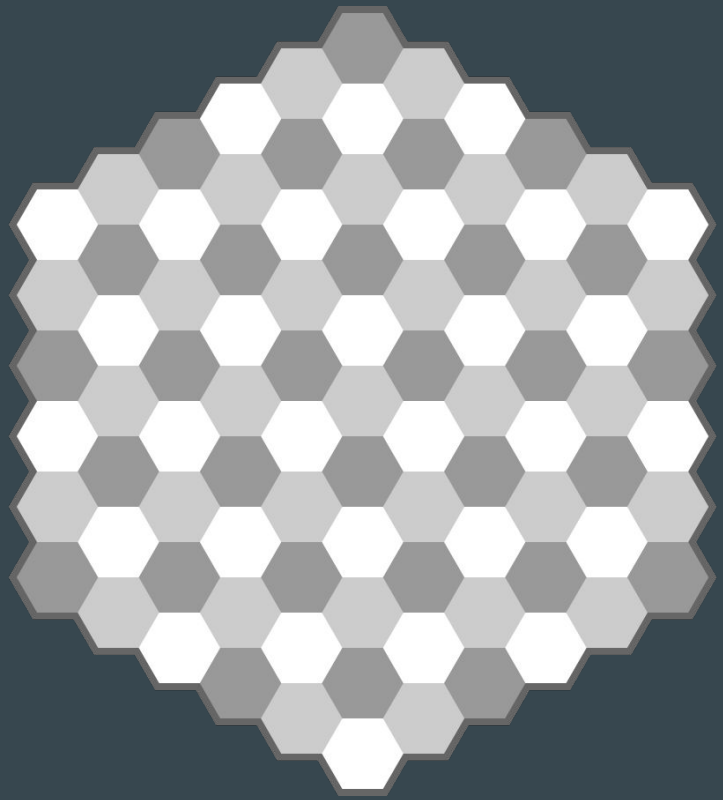
Goal: reach 15 points by purchasing cards

- Players take turns collecting gems and use them to purchase cards worth certain number of points
- Cards provide permanent increase in player's purchasing power
- Bonus points given for collecting card combinations



Future Work

- Expand beyond rectangular/square boards to games with irregular geometry
- Serializing non-grid boards into text without losing geometric relationships is an unsolved representation problem
- Less community data means fewer puzzles, fewer annotated games, and smaller training sets for DSPy optimization



Conclusion

- Explanations turn gameplay into a learning experience and make strategies clear and interpretable
- AI can support human improvement and insight
- Bridges the gap between expert play and beginner learning
- Applies across multiple board games and settings