AI-Enhanced Snakes and Ladders

Submitted By: 22k-4743 (Fatima Salman), 22k-4698 (Hamza salam), 22k-4684(Umer Ahmed)

1. Introduction

The classical board game Snakes and Ladders has been a favorite among players for generations due to its simplicity and luck-based dynamics. However, this project seeks to modernize and intellectually enrich the game by integrating elements of artificial intelligence (AI) and logical reasoning. The central idea behind this reimagination is to transform each player's turn into a learning opportunity by introducing riddles that must be solved before progressing in the game.

In this enhanced version, players—either human or AI—must answer a riddle correctly to roll the dice and move on the board. This simple twist introduces cognitive engagement, strategy, and an educational aspect to an otherwise chance-driven game. The game supports both player-versus-player and player-versus-AI modes, making it versatile for different types of users. The implementation combines artificial intelligence logic, natural language processing, and graphical user interfaces using the Pygame library to deliver an interactive, intelligent, and entertaining experience.

2. Features

The AI-Enhanced Snakes and Ladders game includes a variety of innovative features that contribute to its engaging gameplay and user-friendliness:

2.1 Adaptive Riddle Mechanism

The game features an intelligent riddle system that dynamically adjusts the difficulty level of riddles based on the player's current position or progress. Early in the game, riddles are simpler to help players acclimate, while progressively more challenging riddles are introduced to sustain interest and difficulty. For AI opponents, a Bayesian probability model estimates the AI’s chance of winning and adjusts riddle difficulty accordingly.

2.2 Readable Riddle Display

To ensure accessibility and smooth user experience, each riddle is displayed using a custom text-wrapping mechanism. This ensures that longer riddles are broken into readable lines that fit the display area neatly, avoiding any text overflow or cluttered visuals.

2.3 AI Player with Dynamic Strategy

A key feature of this project is the implementation of a smart AI player. The AI uses a sigmoid-based Bayesian win estimation to evaluate its position relative to the human player. This allows the AI to respond with appropriate difficulty riddles and behave in a way that mimics strategic thinking, offering a challenging experience for human opponents.

2.4 Custom Visual Game Board

The game features a visually appealing, custom-designed board that faithfully represents the traditional layout of Snakes and Ladders. The board includes a background image (board_img) with embedded snakes and ladders, providing clear navigation and aesthetic appeal. Player tokens are distinct and dynamically updated, indicating current positions and transitions effectively.

2.5 Pygame Graphical Interface

The entire application is built using Pygame, a popular game development library for Python. This allows for interactive graphical elements, animations, real-time updates, and a responsive user interface. From drawing the board and tokens to handling user input and game logic, Pygame provides the foundation for a smooth and immersive gameplay experience.

3. Technologies and Libraries

Technology/Library :

Python	Core language for the project logic and structure
pygame	Rendering graphics, handling user input, and drawing the board and UI elements
nltk	Natural Language Toolkit, used to retrieve definitions from WordNet for generating complex riddles
math	Used for sigmoid function in AI's win prediction
random	For dice rolls and randomized riddle selection
time	To manage delays and animation timing

4. Code Overview

4.1 Initialization

Initial configuration sets up window size, grid layout, square size, and frame rate. Colors are predefined using RGB tuples. Pygame's font and display systems are initialized.

4.2 Game Board and Rendering

• board_img: Background image that contains the visual representation of snakes and ladders.

• draw_board(): Draws the board background, player tokens, the question panel, and wrapped riddle text.

4.3 Player and Turn Handling

• players: Stores each player's color and current position.

• animate_move(): Moves a player step-by-step with a delay for animation.

• move_player(): Wrapper to move a player based on dice result.

4.4 Game Logic Functions

• get_coords(position): Converts board position (1-100) to pixel coordinates.

• choose_mode(): Asks the user to select between AI or human opponent.

• main(): Main game loop. Handles input, updates, and win conditions.

4.5 Riddle Mechanism

• easy_riddles: A list of manually added simple riddles.

• hard_riddles: A dynamic list using WordNet definitions as challenging 'riddle-like' prompts.

• ask_question(): Determines riddle difficulty based on game state (Bayesian logic for AI, lead difference for humans).

• check_answer(): Verifies player input against the correct answer.

4.6 AI Logic

• bayesian_win_estimate(ai_pos, human_pos): Sigmoid-based function that estimates AI's chance of winning. This determines riddle difficulty for AI turns.

• ai_turn(): Rolls the dice and moves the AI after solving a riddle based on its win estimate.

4.7 User Interface Helpers

• draw_text_wrapped(): Custom function that wraps long riddle text to fit in a limited width box to improve readability.

• draw_mode_selection(): Displays the mode selection screen at startup.

5. AI Behavior Analysis

The inclusion of an AI player in this version of Snakes and Ladders is more than a simple substitution for a second human player—it introduces dynamic, intelligent behavior designed to offer an engaging and competitive experience. The AI does not simply mimic randomness; instead, its performance and decision-making are influenced by real-time game state evaluations, governed primarily by a Bayesian win probability model.

5.1 Bayesian Win Estimate Function

At the heart of the AI's decision logic lies the bayesian_win_estimate(ai_pos, human_pos) function. This function uses a sigmoid curve to estimate the AI's probability of winning, based on the relative distance between the AI and the human player. The sigmoid function outputs values between 0 and 1, producing a smooth and continuous estimate that increases as the AI gains a lead and decreases when it falls behind. This probabilistic estimate is not just cosmetic—it is tied directly to gameplay decisions.

For instance:

- If the AI is ahead, the win estimate rises, and the game dynamically assigns harder riddles to the AI to balance difficulty.
- If the AI is behind, the system selects easier riddles, giving it a chance to catch up.

This strategy prevents the AI from becoming either too overpowering or too weak, maintaining fair play and challenge throughout the session.

5.2 Riddle Difficulty Adaptation

The AI does not answer riddles in a truly intelligent manner, but the difficulty level of riddles it receives is reflective of its current performance. The sigmoid win estimate plays a key role here. A win estimate above a certain threshold (e.g., 0.6) triggers a hard riddle, while a lower value triggers an easy one. This mimics the idea that a "stronger" AI must work harder to stay ahead—an essential feature in educational or gamified systems to support balanced competition.

5.3 Simulation of Intelligence

Although the AI doesn’t "solve" riddles in the traditional sense, the design creates the illusion of an intelligent agent adapting to the player's skill level. By varying challenge levels and pacing AI progress, the system keeps players invested and emotionally engaged. It also prevents stagnation or boredom, which can occur if the AI wins too easily or loses without resistance.

5.4 Player Experience Consideration

From a player experience standpoint, this AI behavior design offers several benefits:

- Fairness: Players feel they have a genuine chance to win, regardless of initial disadvantage.
- Progressive Challenge: As players improve or gain a lead, the AI becomes more competent, creating a consistently stimulating game.
- Replayability: Varying AI responses across different sessions enhance the replay value, as outcomes are no longer purely luck-based.

5.5 Analytical Observations

During testing, the AI behavior was monitored across multiple game sessions:

- When trailing by more than 15 positions, the AI was more likely to receive easy riddles, solving them quickly and closing the gap.
- When significantly ahead, the increased difficulty of riddles often resulted in AI errors or slower progress, allowing the human player to remain competitive.

This design reinforces adaptive difficulty not only as a feature but as a philosophy: a way of ensuring that players, regardless of skill, remain challenged but not overwhelmed.

6. Conclusion

This project successfully demonstrates how a traditional game like Snakes and Ladders can be transformed into a modern, intelligent, and educational experience using artificial intelligence and riddles. The adaptive difficulty mechanism, AI opponent, and graphical enhancements make the game both entertaining and mentally stimulating.

Its modular codebase written in Python, along with integration of libraries like Pygame and NLTK, ensures that the project is highly extensible and customizable. It holds potential for various applications, including educational gaming, classroom activities, or even casual play for children and adults alike.