Advanced Scrabble AI

Asghar Ali (22K-4415)
Muhammad Bilal (22K-4242)
Hafiz Abdullah (22K-4489)

Course: AI
Instructor: Abdullah Yaqoob

Abstract

This project presents an "Advanced Scrabble AI," an enhanced version of the classic Scrabble word game. The primary goal was to develop a competitive AI opponent capable of strategic gameplay, integrated with novel game mechanics to increase complexity and player engagement. The AI utilizes the Minimax algorithm with Alpha-Beta pruning for intelligent move selection. The game introduces features such as power tiles (e.g., Double Turn) and dynamic player objectives, offering a fresh take on the traditional Scrabble experience. The system is architected with a Python FastAPI backend for game logic and AI processing, and a React JS frontend for an interactive user interface.

1 Introduction

1.1 Project Scope and Objectives

The core objectives outlined for this project were:

- **Develop a Competitive AI:** Implement an AI using the Minimax algorithm with Alpha-Beta Pruning to make intelligent move selections based on maximizing its score and considering the game state.
- **Introduce Gameplay Variations:**
  - Implement **Power Tiles** to introduce special effects during gameplay.
  - Implement **Dynamic Objectives** to provide players with secret missions for bonus points, adding another strategic dimension.
- **Implement Robust Game Logic:** Create a backend system to manage game state, validate moves, calculate scores, and enforce all game rules, including the new variations.
- **Develop an Interactive User Interface:** Build a frontend that allows a human player to interact with the game and play against the AI.
- **Heuristic Design:** Implement heuristics for the AI to evaluate board states, considering factors like score potential and rack balance.

2 Game Description

2.1 Core Scrabble Gameplay

The foundation of our project is the classic Scrabble game:

- Players have a rack of letter tiles.
- Words are formed on a 15x15 grid, connecting to previously played words.
- The first word must cover the center square.
- Tiles have point values, and premium squares (Double/Triple Letter, Double/Triple Word) multiply scores.
- Players draw new tiles to replenish their rack to seven tiles.
- The game ends when all tiles are drawn and one player empties their rack, or after a set number of consecutive passes.

2.2 Implemented Innovations

To enhance the traditional gameplay, the following innovations have been successfully implemented:

- **Power Tiles:**
  - **Double Turn (D*):** A special tile that, when played as part of a valid word, grants the player an immediate extra turn. This adds a significant tactical advantage.
- **Dynamic Objectives:**
  - Each player (human and AI) is assigned a secret objective at the start of the game (e.g., "Score 30+ points in a single turn," "Play a word using Q, Z, X, or J," "Form a 7-letter word," "Play a tile on a corner square").
  - Completing an objective awards the player bonus points, encouraging diverse playstyles beyond simple score maximization on every turn.

3 System Architecture

The project is a full-stack web application with a clear separation of concerns between the backend and frontend.

3.1 Backend (FastAPI - Python)

The backend is responsible for all game logic, AI decision-making, and state management.

- **Language:** Python 3+
- **Framework:** FastAPI for creating robust and efficient RESTful APIs.
- **Key Modules:**
  - `main.py`: Sets up the FastAPI application, defines API endpoints, and manages the global game instance.
  - `game_logic/board.py`: Contains the `Board` class, which encapsulates all rules, board state, tile bag, player racks, score calculation, move validation, and objective tracking.
  - `game_logic/ai_player.py`: Houses the `AIPlayer` class, implementing the Minimax algorithm with Alpha-Beta pruning and the heuristic evaluation function.
  - `game_logic/constants.py`: Stores game constants like letter scores, power tile definitions, and objective types.
  - `game_logic/utils.py`: Includes utility functions for dictionary loading, word validation, and finding anchor squares.
  - `models.py`: Defines Pydantic models for API request and response data validation and serialization.

3.2 Frontend (React - JavaScript)

The frontend provides the graphical user interface for players to interact with the game.

- **Language:** JavaScript
- **Framework/Library:** React
- **Key Components:**

  - `App.jsx`: Main application component, handles routing and global game context.
  - `GameProvider` (within `App.jsx`): Manages frontend game state, API communication, and UI logic (tile selection, placement).
  - `Board.jsx`: Renders the Scrabble board, player rack, scoreboard, objectives, and game controls.
- **Styling:** CSS (`App.css`, `boardstyles.css`) for visual presentation and responsiveness.

3.3 API Design

A RESTful API facilitates communication between the frontend and backend:

- `GET /api/game/start`: Initializes a new game.
- `POST /api/game/move`: Submits a human player's move.
- `POST /api/game/pass`: Allows a human player to pass their turn.
- `GET /api/game/state`: Retrieves the current game state.

Data is exchanged in JSON format, validated using Pydantic models on the backend.

4 AI Engine Design and Implementation

4.1 Core Algorithm: Minimax with Alpha-Beta Pruning

The AI opponent's intelligence is driven by the Minimax algorithm:

- **Minimax:** Explores the game tree by assuming both players play optimally. The AI (maximizer) tries to maximize its score, while assuming the opponent (minimizer) will try to minimize the AI's score.
- **Alpha-Beta Pruning:** An optimization technique for Minimax that significantly reduces the number of nodes evaluated by pruning branches that cannot influence the final decision.
- **Search Depth:** The AI searches to a predefined depth (`MAX_DEPTH` = 1 in the current configuration) to evaluate potential future game states.

5 Future Work and Potential Enhancements

- **Enhanced AI Heuristics:** Incorporate board control metrics, defensive play considerations, and "leave valuation."
- **Increased AI Search Depth/Efficiency:** Optimize further or explore techniques like transposition tables.
- **User Accounts and Game History.**
- **Multiplayer (Human vs. Human).**
- **Advanced AI Techniques:** Explore Monte Carlo Tree Search (MCTS) or Reinforcement Learning.

6 Conclusion

The "Advanced Scrabble AI" project successfully developed a functional and engaging Scrabble game featuring a competitive AI opponent and novel gameplay elements like power tiles and dynamic objectives. The AI, based on Minimax with Alpha-Beta pruning, exhibits strategic decision-making. The project provided valuable experience in AI, full-stack development, and game logic design. While some ambitious features were deferred, the core objectives were met, resulting in a robust application with a modular architecture amenable to future enhancements.