Descripción de Clases para el Proyecto de Tik Tak Toe

Clase AVLNode

Representa un nodo en el árbol AVL que almacena el estado del tablero y los valores Q asociados.

Atributos:

- board_state: Estado del tablero representado como una tupla.
- value_q: Diccionario que mapea movimientos a valores Q.
- height: Altura del nodo en el árbol AVL.
- left: Nodo hijo izquierdo.
- right: Nodo hijo derecho.

Métodos:

- __init__(self, board_state, value_q): Constructor de la clase que inicializa los atributos del nodo.

Clase AVLTree

Maneja las operaciones del árbol AVL para almacenar y gestionar los estados del tablero y sus valores Q.

Atributos:

- root: Raíz del árbol AVL.

Métodos:

- __init__(self): Constructor de la clase que inicializa la raíz del árbol.
- insert(self, node, board_state, value_q): Inserta un nuevo nodo en el árbol AVL.
- get_height(self, node): Obtiene la altura de un nodo.

- get_balance(self, node): Calcula el balance de un nodo.

- rotate_left(self, z): Realiza una rotación a la izquierda.

- rotate_right(self, z): Realiza una rotación a la derecha.

- search(self, node, board_state): Busca un nodo por su estado del tablero.

- get_all_nodes(self): Obtiene todos los nodos del árbol en recorrido in-order.

- _inorder_traverse(self, node, nodes): Recorre el árbol en in-order y almacena los nodos en una lista.

- visualize_tree(self, filename='avl_tree'): Genera una visualización del árbol AVL usando Graphviz.

- _add_nodes(self, dot, node): Añade nodos al gráfico de Graphviz.

- _add_edges(self, dot, node): Añade aristas al gráfico de Graphviz.

- delete_node(self, node, board_state): Elimina un nodo del árbol AVL.

- get_min_value_node(self, node): Obtiene el nodo con el valor mínimo en un subárbol.

- remove_duplicates(self): Elimina nodos duplicados en el árbol.

Clase BoardManager

Gestiona la interfaz gráfica del juego y la interacción con el usuario.

Atributos:

- root: Ventana principal de la interfaz gráfica.

- avl_tree: Instancia del árbol AVL para almacenar estados del juego.

- gameUtilities: Utilidades del juego para tareas auxiliares.

- machineIa: Instancia de la IA del juego.

- buttons: Lista de botones que representan las celdas del tablero.

- score_x: Puntuación del jugador 'X'.

- score_o: Puntuación del jugador 'O'.

- draws: Número de empates.

- turn: Indica el turno actual ('X' o 'O').

- score_label: Etiqueta para mostrar la puntuación en la interfaz gráfica.

Métodos:

- __init__(self): Constructor de la clase que inicializa la interfaz gráfica.

- init_Ia(self, new_self): Inicializa la IA del juego.

- create_widgets(self): Crea los widgets de la interfaz gráfica.

- reset_game(self, silent=False): Reinicia el estado del juego.

- update_score(self, winner): Actualiza el marcador del juego.

- update_scores(self): Muestra el marcador en la interfaz gráfica.

- winner(self): Determina el ganador del juego.

- create_menu(self): Crea el menú interactivo.

- show_history(self): Muestra el historial de partidas.

- show_avl_tree(self): Muestra la visualización del árbol AVL.

- show_group_information(self): Muestra la información de los integrantes del grupo.

- on_button_press(self, index, pvpMode=True): Maneja la acción al presionar un botón del tablero.

- get_board_state(self): Obtiene el estado actual del tablero como una tupla.

- ask_training_games(self): Pide al usuario el número de partidas para entrenar la IA.

Clase MachineIa

Implementa la lógica de la IA del juego, incluyendo el aprendizaje supervisado.

Atributos:

- boardContext: Contexto del tablero del juego.

- avl_tree: Instancia del árbol AVL para almacenar estados del juego.

- gameUtilities: Utilidades del juego para tareas auxiliares.

Métodos:

- __init__(self, boardContext): Constructor de la clase que inicializa la IA.

- machine_move(self, pvpMode=False): Realiza el movimiento de la máquina.

- execute_move(self, index, player, pvpMode=True): Ejecuta un movimiento en el tablero.

- update_q_values(self, state, action_index, reward, is_diagonal_move=False, blocked_opponent=False, gamma=0.9): Actualiza los valores Q de un estado.

- block_opponent_win(self, empty_indices, current_state): Intenta bloquear una jugada ganadora del oponente.

- evaluate_diagonal(self, state, index): Evalúa si un movimiento es diagonal y si bloquea al oponente.

- choose_best_move(self, state, possible_moves): Elige el mejor movimiento basado en los valores Q.

- evaluate_move_result(self, index): Evalúa el resultado de un movimiento.

- train_model(self, N=100): Entrena el modelo de la IA con N partidas.

- simulate_game(self, use_x): Simula un juego para entrenar la IA.

- get_best_q_value(self): Obtiene el mejor valor Q en el árbol AVL.

Clase TicTacToeApp

Gestiona la aplicación principal del juego.

Atributos:

- board: Instancia de BoardManager para gestionar la interfaz gráfica.

Métodos:

- __init__(self): Constructor de la clase que inicializa la aplicación.

- mainloop(self): Inicia el bucle principal de la interfaz gráfica.

Clase GameUtilities

Provee funciones auxiliares para el juego.

Atributos:

- boardContext: Contexto del tablero del juego.

Métodos:

- __init__(self, boardContext): Constructor de la clase que inicializa las utilidades.

- save_screenshot(self): Guarda una captura de pantalla del estado actual del juego.