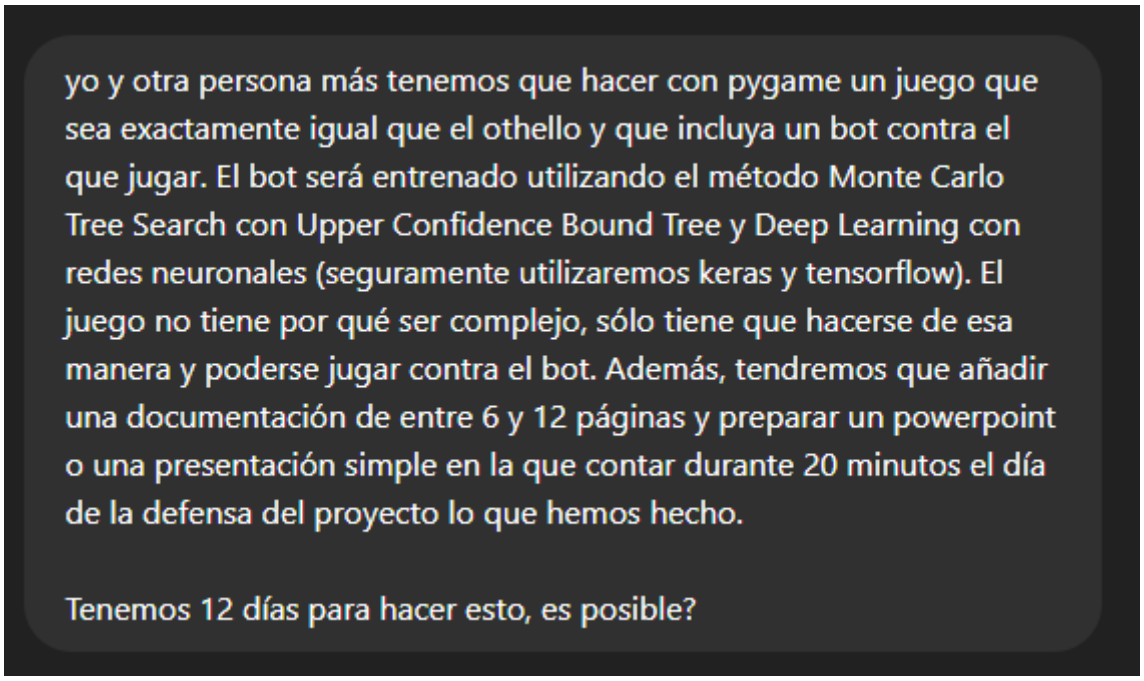
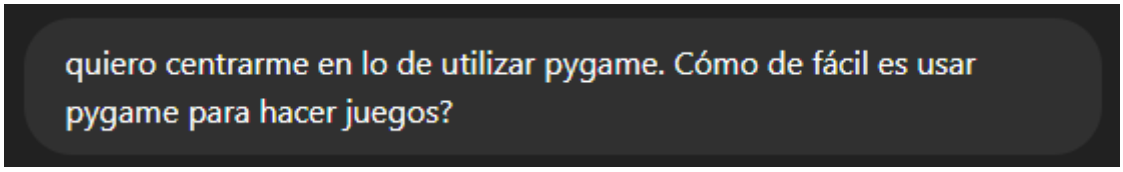
Uso de IA generativa

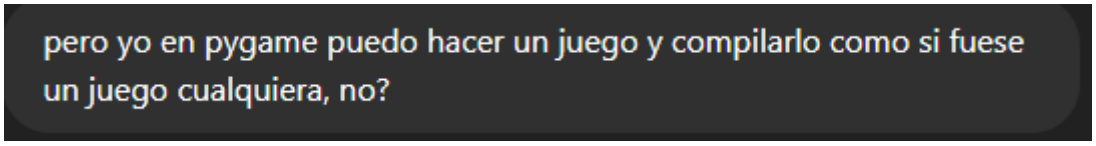
Por parte del alumno Eloy Sancho Cebrero, se utilizó exclusivamente ChatGPT.

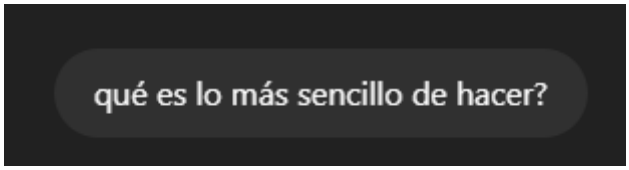
Primero se quiso poner en contexto a la IA. Se le preguntó sobre su concepción del tamaño del proyecto, pero esto no fue lo realmente importante.



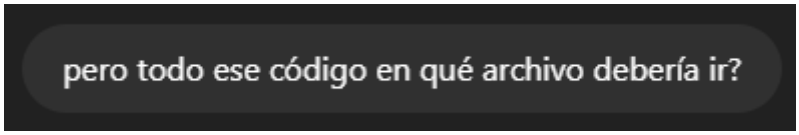
Se le preguntó sobre pygame para entender el funcionamiento de la librería.

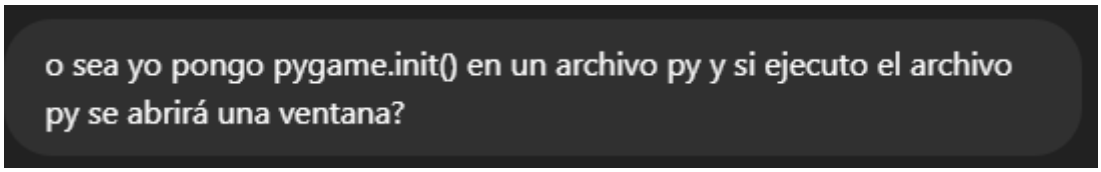


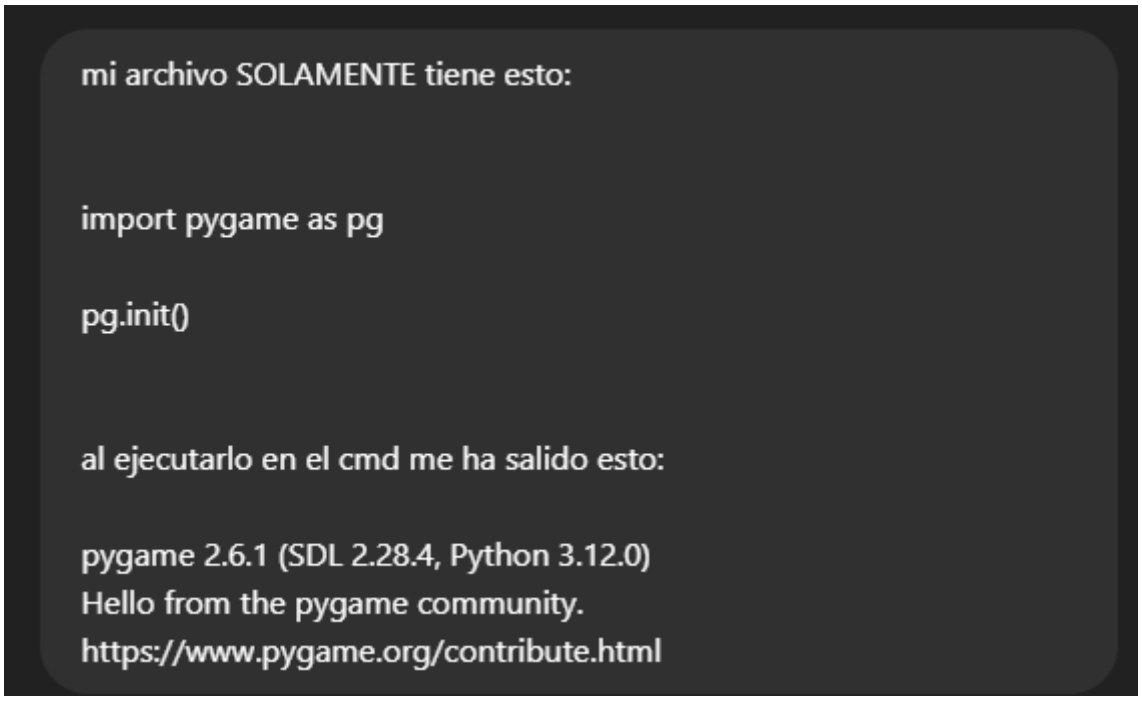




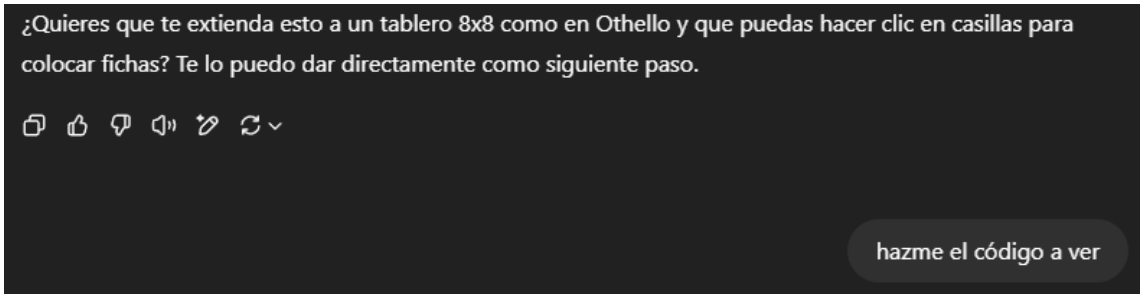
Dado código de ejemplo muy simple, se le preguntó en qué lugar escribirlo.



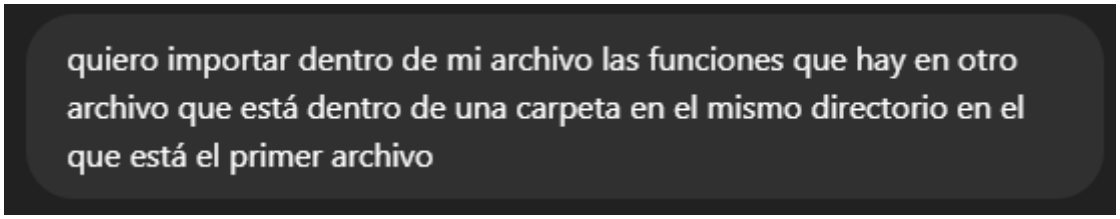




Se le pidió un código base que pudiese crear un juego de Otelo sencillo.



El código proporcionado fue una buena base, pero se quiso ampliar y modificar pues dicho código sólo colocaba fichas en cualquier casilla, sin implementar restricciones de Otelo. Para poder separar las funciones del script principal y añadir nuevas se hizo uso de la IA.



También se cuestionó sobre la importancia de pycache en el repositorio.

al ejecutar el juego se ha creado una carpeta pycache, debería incluirla en el gitignore?

Se le hicieron preguntas sobre el código proporcionado.

no entiendo muy bien esto.
1) Comprendo que pg.MOUSEBUTTONDOWN se utiliza para detectar un clic del ratón, pero "evento.button == 1" qué significa? Se refiere a que es el clic izquierdo del ratón?
2) qué representa evento.pos? es decir, qué valores almacenará en x e y? y por qué esos valores se utilizan para detectar la columna y la fila y, adicionalmente, se realiza esa operación que no conozco con el tamaño de las casillas?

↪ pg.draw.rect(pantalla, COLOR_LINEAS, (x, y, TAM_CASILLA, TAM_CASILLA), 1)

esto tampoco logro entenderlo. Estás dibujando las rectas que separan las casillas o sólo estás dibujando un cuadrado que rodea la casilla?

Se usó como apoyo para implementar las reglas de Otelo.

Cuál sería la forma óptima de comprobar que se coloca una ficha en un sitio adecuado y que las fichas que hay en medio se cambian? Me basta con una explicación concreta de este asunto incluyendo ejemplos si es necesario, no quiero que me des el código de todo el juego hecho con estas restricciones

La función obtener_fichas_a_voltear recibirá como parámetro el tablero (la matriz)
hazme un while que pueda ser utilizado como base para recorrer cada dirección. El while debe sumar 1 a la fila o columna y parar cuando el parámetro turno sea distinto al encontrado en ese lugar de la matriz se detenga

cómo integrar la lista de direcciones con ese while?

qué problema detectas en mi función?:

```
def obtener_fichas_a_voltear(tablero, fila, col, turno):
    '''
    '''
    direcciones = [(-1,-1),(-1, 0),(-1, 1),
                   ( 0,-1)       ,( 0, 1),
                   ( 1,-1),( 1, 0),( 1, 1)]
    res = []

    for df, dc in direcciones:
        nf = fila + df
        nc = col + dc

        while 0 <= nf < 8 and 0 <= nc < 8:
            nueva_ficha = tablero[nf][nc]

            if nueva_ficha != turno:
                res.append((nf, nc))
            else:
                break

    return res
```

como podría comprobar que un jugador no puede realizar movimientos y, por ende, su turno debe pasar?

Puedes darme una secuencia de movimientos con la que comprobar sencillamente lo que acabo de implementar?

Después se le preguntó sobre los siguientes pasos a seguir en la creación del proyecto, así como para aclarar algunas partes del funcionamiento de MCTS. Incluso pudo proporcionar un código base para la clase Nodo.

En mis instrucciones, se me indica que para poder entrenar una red neuronal que ayude a evaluar posiciones en el juego, es necesario contar con un agente que pueda jugar de forma básica. Este agente debe estar implementado usando MCTS con UCT.
Primero tendré que generar partidas automáticas haciendo que la máquina juegue partidas contra sí misma, en cada partida se registran todos los estados intermedios.

Eso implica que primero será necesario crear el MCTS?

En la búsqueda adversaria y en el contexto del Otelo:
dado un estado s0, el algoritmo de montecarlo devolverá la mejor acción a tomar desde ese estado. En en este contexto del otelo, un estado es una matriz 8x8 que representa el tablero en ese momento. Con el algoritmo de Monte Carlo Tree Search, será necesario crear un agente que juegue partidas y genere datos que serán usados por una red neuronal. Esos datos serán los estados (tableros 8x8), junto con una etiqueta que determina si la partida simulada partiendo de ese estado acabó en derrota (-1), empate (0) o victoria (+1). La cuestión es, ¿cómo el agente generaría esos datos si, el algoritmo de Monte Carlo, lo único que hace es recibir un estado (una matriz 8x8 en nuestro caso) y devolver una acción a tomar (en nuestro caso, colocar una ficha en una casilla)?

pero hay generar todas las posibles matrices del otelo y utilizar la función mcts con cada una de esas matrices, obteniendo una etiqueta? Si es así, para qué exactamente el algoritmo devuelve una acción?

es decir, le doy al agente el estado inicial (en nuestro caso el estado inicial puede ser uno de los posibles movimientos del negro, pues el bot podría ser el blanco). A partir de ahí, usa la función mcts. (¿Podría devolver directamente el estado hijo en lugar de la acción? Creo que sería más sencillo de implementar que devolver la acción como tal). Una vez tiene el estado hijo, coloca la ficha para que se de ese estado. Después, para cada estado de los posibles (es decir, los tableros resultantes de cada uno de los posibles movimientos del jugador) vuelve a hacer mcts. Y lo repite hasta que se terminen las partidas.

Lo que no sé es cómo implementar eso en código, principalmente la parte en la que la partida se ramifica en varias partidas en las que el jugador rival al bot ha tomado una de las distintas acciones posibles.

por ahora tengo esto, qué mejoras habría que hacer?

```
def mcts(tablero, turno):
    '''
    '''

    i = 0
    while i <= 500:
        tablero_nuevo = tree_policy(tablero)
        reward = default_policy(tablero)
        backup(tablero_nuevo, reward)
    return mejor_movimiento(tablero, 0)
```

el hecho de que Q(s) sea "la recompensa total de todas las simulaciones que han pasado por el estado s" no cambia el hecho de que mis estados sólo estén etiquetados con -1,0 o 1 no? porque Q(s) no cambia etiquetas, sólo es un valor utilizado por UCT

Varios apuntes sobre la clase Nodo y lo que has escrito sobre mcts:

1) Para qué se utiliza el atributo "movimiento" de la clase Nodo?
2) no es necesario, en mcts, especificar los hijos que tiene el nodo root? Para ello tengo la función movimientos_disponibles(tablero, turno) que devuelve una lista de tuplas con la fila y la columna de cada uno de los lugares en los que el jugador puede poner una ficha

Respecto al punto 2:
El pseudocódigo de tree_policy lo primero que hace es comprobar que el nodo no es terminal y, para ello, tengo pensado comprobar que no tenga hijos disponibles. Sin embargo, eso no es suficiente, puesto que en caso de que un nodo no tenga hijos, no quiere decir que la partida haya acabado, puesto que puede ser que ese jugador no tenga movimientos disponibles, habría que comprobar que el otro jugador (es decir, el siguiente nodo, siendo "siguiente nodo" un nodo en el que el tablero es EL MISMO, pero el turno es el del otro jugador) tampoco tenga movimientos disponibles.

y cómo comprobar si un nodo no está expandido? o mejor dicho, qué significa que un nodo no está expandido? que todavía no tiene todos sus posibles hijos en su lista de hijos (su atributo "hijos")?

tengo esta función expand, es correcta?

```
def expand(nodo):
    '''
    '''

    movimiento = nodo.movimientos_por_hacer.pop()
    nuevo_tablero = nodo.tablero
    nuevo_tablero[movimiento[0]][movimiento[1]] = nodo.turno
    hijo = Nodo(nuevo_tablero, 3 - nodo.turno, padre=nodo)
    nodo.hijos.append(hijo)
    return hijo
```

tengo esta función tree_policy

```
def tree_policy(nodo):
    '''
    '''

    while no_terminal(nodo):
        if nodo.movimientos_por_hacer:
            return expand(nodo)
        else:
            nodo = mejor_hijo(nodo, Cp)

    return nodo
```

ahora quiero elegir aleatoriamente una acción de movimientos_disponibles

Si los estados deben tener etiquetas (recompensas) y en la función default policy es necesario devolver la recompensa del estado siguiente escogido al azar, ¿cómo relacionar cada estado con una etiqueta, debería hacerlo en la clase Nodo? Esta es la función default policy que tengo hecha (aún incompleta, falta el return):

```
def default_policy(nodo):
    '''
    '''

    while no_terminal(nodo.estado, nodo.turno):
        acciones = movimientos_disponibles(nodo.estado, nodo.turno)
        if acciones:
            movimiento = random.choice(acciones)
            nuevo_tablero = copy.deepcopy(nodo.tablero)
            nuevo_tablero[movimiento[0]][movimiento[1]] = nodo.turno
        else:
            nuevo_tablero = copy.deepcopy(nodo.tablero)
```

esta función backup entiendo que encaja con la función mcts que tengo verdad? Entiendo que backup va a ir cambiando el nodo que se maneja dentro de mcts hasta llegar al primer nodo. Después entiendo que mcts hará más iteraciones hasta que llegue a las 500 iteraciones (incluida)

Función backup:

```
def backup(nodo, recompensa):
    '''
    '''

    while nodo is not None:
        nodo.n = nodo.n + 1
        nodo.q = nodo.q + recompensa
        nodo = nodo.padre
```

Función mcts:

```
def mcts(tablero, turno, iteraciones=500):


    root = Nodo(tablero, turno)
    i = 0
    while i <= iteraciones:
        nodo = tree_policy(root)
        recompensa = default_policy(nodo)
        backup(nodo, recompensa)
        i += 1

    return mejor_movimiento(root)
```

La función mejor movimiento aún no está implementada, pero creo que puede sobrar. Realmente lo que quiero para seguir el pseudocódigo del paper es devolver la acción que lleva al nodo root a su mejor hijo ("return a(BESTCHILD(v0,0))", siendo 0 el valor del parámetro c en la función "mejor_hijo"). Aquí está el código de "mejor_hijo":

```
def mejor_hijo(nodo, c):

    elegido = None
    uct = None
    for h in nodo.hijos:
        uct_actual = (h.q/h.n) + c*sqrt((2*log(nodo.n))/h.n)
        if uct == None or uct_actual > uct:
            uct = uct_actual
            elegido = h

    return elegido
```

1) es esta función correcta?
2) habría forma de aprovechar esta función para obtener el return deseado?

Una vez hecha la implementación del motor MCTS, se le pasó la implementación y se le hicieron preguntas al respecto. La IA fue capaz de generar un código base para generar el csv de partidas que utilizaría la red neuronal para su entrenamiento.

1) Hay algo que corregir en esta implementación?
2) por dónde debería empezar para probar este agente y, lo que es más importante, generar los datos que serán usados por una red neuronal? Mi objetivo por el momento es generar los datos a modo de lista de tuplas en las que cada tupla tiene dos elementos: una matriz 8x8 (el tablero que representa el estado en cuestión) y su respectiva etiqueta (-1,0,+1).

creo que en mi caso debería generar los datos a modo de csv, pues es lo que se adapta al contenido de redes neuronales que he dado en la asignatura. En ese caso, cada fila del csv tendría el tablero y la etiqueta

cómo importo a agente_mcts.py las funciones de mcts.py si mcts.py está en la misma carpeta que agente_mcts.py?

para qué sirve el parámetro jugador de la funcion generar csv?

Ahora necesito, en ese mismo archivo .py, crear la función main para que, al ser ejecutado el archivo, se genere el csv deseado en el que cada fila es un tablero otelo (64 números que representan lo que hay en cada casilla, 0:ninguna, 1:blanca, 2:negra) y una etiqueta que indica si la partida a la que pertenece ese estado perdió (-1), empató (0) o ganó (1)

me ha surgido un error de importación en:

from utiles.fichas import *

puede deberse a que utiles no está dentro de la carpeta en la que se encuentra este script.
La estructura de carpetas es:

> otelo (carpeta principal del repositorio):
  > mcts
  > red_neuronal
  > utiles
  .gitignore
  otelo.py
  README.md

cuánto debería tardar si he especificado 100 iteraciones?

el csv que se genera está vacío y ademas el script no termina nunca. a qué se debe?: esto es lo que hay en agente_mcts (he puesto 1 partida y 1 iteración por ahora sólo para comprobar que se generan los datos):

(...)

con 10 partidas y 20 iteraciones me ha generado tableros en los que las etiquetas son 0. Eso es normal?

he vuelto a generarlo y las 10 partidas, según los datos generados, han acabado en empate. Creo que no se añaden las etiquetas correctamente. Por ejemplo: esta es la última fila del csv:
1,2,1,2,1,0,1,2,2,1,2,1,2,1,2,1,1,2,1,2,1,2,1,2,2,1,2,1,2,1,2,1,2,2,1,2,1,2,1,2,1,1,2,1,2,1,2,1,2,2,1,2,1,2,1,2,1,2,1,2,1,0
1) Hay 65 números?
2) En caso de que haya 65 números, ¿verdaderamente hay el mismo número de 2 que de 1 (sin contar con el número final)?

esta es la función obtener ganador:

```
def obtener_ganador(tablero):
    '''
    '''

    contador_negras = 0
    contador_blancas = 0
    for fila in range(8):
        for col in range(8):
            if tablero[fila][col] == 1:
                contador_blancas += 1
            if tablero[fila][col] == 2:
                contador_negras += 1

    if contador_negras > contador_blancas:
        return 2
    elif contador_blancas > contador_negras:
        return 1
    elif contador_blancas == contador_negras:
        return 0
```

no_terminal hace esto:

```
def no_terminal(tablero, turno):
    '''
    '''

    movimientos_propios = movimientos_disponibles(tablero, turno)
    # "3 - turno" alterna el turno. Si turno es 2 (negras), 3-2=1 (blancas), si turno es 1, 3-1=2
    movimientos_oponente = movimientos_disponibles(tablero, 3 - turno)

    return movimientos_propios or movimientos_oponente
```

esta es movimientos_disponibles:

```
def movimientos_disponibles(tablero, turno):
    '''
    Esta función devuelve una lista con las coordenadas de los lugares en los que el jugador puede colocar una ficha.

    PARÁMETROS
    - tablero: Matriz 8x8 con los valores que indican qué ficha hay en cada casilla.
    - turno: Número que indica qué jugador está en su turno (0: ninguno, 1: blancas, 2: negras).
    '''

    res = []
    for fila in range(8):
        for col in range(8):
            if tablero[fila][col] == 0:
                fichas = obtener_fichas_a_voltear(tablero, fila, col, turno)
                if fichas:
                    res.append((fila,col))

    return res
```

en la función expand debería hacer lo mismo no? Pues el nuevo tablero también ha de tener las fichas volteadas:

```
def expand(nodo):
    ...
    ...

    movimiento = nodo.movimientos_por_hacer.pop()
    nuevo_tablero = copy.deepcopy(nodo.estado)
    nuevo_tablero[movimiento[0]][movimiento[1]] = nodo.turno
    hijo = Nodo(nuevo_tablero, 3 - nodo.turno, padre=nodo,
movimiento=movimiento)
    nodo.hijos.append(hijo)

    return hijo
```

tras los cambios aplicados, la función tree_policy genera un error en el while no_terminal(nodo.estado, nodo.turno)

se debe a que ha detectado que el nodo es None. Esta es mi función tree_policy:

(...)

sigue dando el mismo error, mejor_hijo es esto:

```
def mejor_hijo(nodo, c):

    elegido = None
    uct = None
    for h in nodo.hijos:
        if h.n == 0:
            uct_actual = float('inf')
        else:
            uct_actual = (h.q/h.n) + c*sqrt((2*log(nodo.n))/h.n)
        if uct == None or uct_actual > uct:
            uct = uct_actual
            elegido = h

    return elegido
```

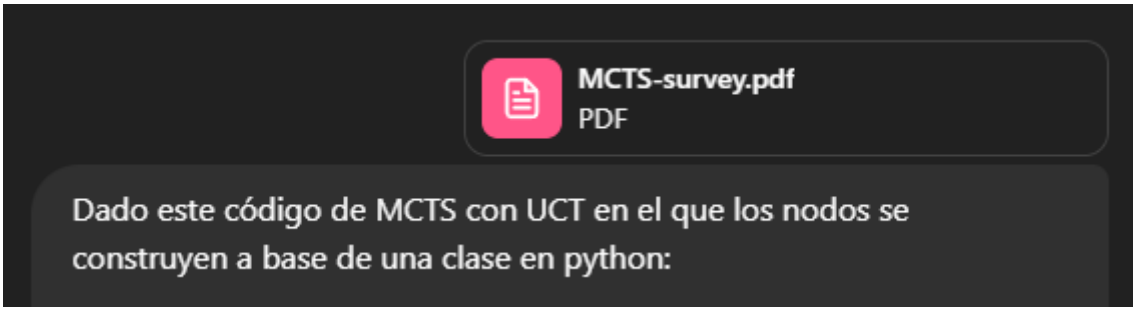
ahora no termina de generarse el csv porque hay un bucle infinito

Tenemos la siguiente función en el contexto de Monte Carlo Tree Search con uso de la ecuación UCT:

(...)

En el momento en el que se comprueba si n es 0, se establece uct como infinito. Esto es matemáticamente correcto, pues se dividirá por 0 en la ecuación y dará infinito. Ahora bien, esto hará que un nodo que ha sido visitado 0 veces de infinito. Es decir, el primer nodo que de infinito en la ecuación UCT, será el elegido. Entiendo que esto es correcto en el algoritmo MCTS, pues queremos visitar los nodos que no hemos visitado nunca, verdad?

Una vez arreglados los problemas, se uso la IA para aclarar una duda con respecto al correcto funcionamiento del código respecto al pseudocódigo del algoritmo.



(...)

Tengo serias dudas de si las variables q y n se quedan correctamente registradas en las iteraciones de MCTS. Entiendo que en una iteración de MCTS se crea un nodo hijo del nodo raíz, se simula la partida y se calcula la recompensa y se retropropaga la recompensa. Pero, realmente, en la siguiente iteración, el nodo hijo del raíz, no será el mismo, sino que se construirá otro nodo hijo y el proceso volverá a cero.

El pseudocódigo en el que se basa la función mcts es el siguiente:

```
function UCTSEARCH(s0)
create root node v0 with state s0
    while within computational budget do
      vl
      TREEPOLICY(v0)
      DEFAULTPOLICY(s(vl))
      BACKUP(vl, )
    return a(BESTCHILD(v0,0))
```

Algo me hace pensar que el "while within computational budget do" no es equivalente al mecanismo de iteraciones de mi función, pues en cada iteración, las variables se definen de nuevo y los nodos hijos también.
Te adjunto el documento de MCTS por si lo necesitas para analizar mejor el funcionamiento del código y poder propocionar una solución más eficiente y que más se adapte al planteamiento del paper adjunto en este mensaje.
Si necesitas las funciones adicionales para hacer un mejor análisis házmelo saber, pero no lo veo necesario por ahora.

No me convence la explicación. Esto es lo que creo que hace el código:

Dentro del bucle se define el nodo hijo según la función tree_policy (que expandirá el nodo raíz y devolverá el mejor hijo). Eso implica que en la siguiente iteración, volverá a expandir el nodo raíz desde 0 y volverá a elegir el mejor hijo. Al expandir el nodo raíz nuevamente, volverá a construir esos hijos y, por tanto, los valores n y q resultantes de la iteración anterior, no habrán servido para nada, puesto que en esta iteración antigua no se guardaron los nodos construidos en ningún sitio. El único nodo que persiste a las iteraciones es el nodo raíz, qué si verá su n y su q modificada, pero no sus hijos.

Es cierto que el código hace esto? En caso de que mi análisis sea acertado, qué podría hacer para resolver este problema?

entonces, gracias al atributo "hijos", del nodo raíz, los hijos quedan guardados en la memoria y no se vuelven a construir. Por ejemplo, en la segunda iteración, al estar ya expandido el nodo raíz, tree_policy ya no usará la función expand porque los hijos ya existen y ya están almacenados en ese atributo del nodo raíz.

Posteriormente, se utilizó la IA para poder generar la documentación haciendo uso de LaTeX.

Tengo que crear documentación utilizando el sistema LaTeX y siguiendo el formato IEEE Conference Proceedings. Mi sistema operativo es Windows y se me dan las opciones MikTeX y TeXLive. Quiero instalar la opción más sencilla (tengo entendido que es MikTeX) y que se pueda usar perfectamente para seguir el formato IEEE indicado anteriormente.

Los ejemplos de la web de LaTeX no me sirven, necesito una plantilla desde la que partir para hacer mi documentación. A ser posible, siguiendo el formato IEEE Conference Proceedings

En caso de markdown cuando una línea se sale del espacio que ve tu editor de texto, el visualizador te muestra la línea justo debajo, pero el editor de LaTeX de VSCode no lo hace. Hay forma de forzarlo a hacerlo?

cómo iniciar el engine de miktex?

entonces, qué debería hacer con los errores de VSCode de tipo:

The log file hopefully contains the information to get MiKTeX going again:

C:\Users\----\AppData\Local\MiKTeX\miktex\log\latexmk.log

For more information, visit: https://miktex.org/kb/fix-script-engine-not-found

Aprendiendo_a_jugar_a_Otelo-2.pdf
PDF

Dado este documento que describe las instrucciones a seguir y evaluación de un proyecto a implementar, necesito que solamente te limites a escribir el "abstract" a incluir en la documentación de dicho proyecto, documentación realizada a modo de paper científico tal y como se indica en estas instrucciones.

cómo traduzco la lista de elementos tal y como lo especifica el formato IEEE a LaTeX?

este texto dentro de una lista me ha dado el error "Missing $ inserted." (aunque no te lo muestre, arriba la lista de items empieza y tiene otros elementos):

\item \textbf{Movimientos por hacer}. Almacena el conjunto de movimientos (representados mediante la tupla: (fila, columna)) disponibles para el jugador que puede colocar ficha (según el atributo "turno"). Es perfectamente posible que el jugador no tenga movimientos disponibles y la lista de movimientos por hacer esté vacía. Este atributo siempre se inicializa mediante la función movimientos_disponibles que devuelve lo deseado a partir del estado del tablero (el atributo "estado") y el turno del jugador (el atributo "turno").
    \item \textbf{Movimiento}. Almacena el movimiento (representado mediante la tupla: (fila, columna)) que ha dado lugar al estado del nodo en cuestión. Este movimiento es siempre nulo en el estado inicial de la partida, pues no se ha realizado ningún movimiento previo a dicho estado. Por ello, su valor por defecto es nulo y debe ser especificado si se quiere construir un Nodo que sí tenga movimiento "creador".
\end{itemize}

da el mismo error:

\item \textbf{Movimientos por hacer}. Almacena el conjunto de movimientos \texttt{(representados mediante la tupla: fila, columna)} disponibles para el jugador que puede colocar ficha \texttt{(según el atributo "turno")}. Es perfectamente posible que el jugador no tenga movimientos disponibles y la lista de movimientos por hacer esté vacía. Este atributo siempre se inicializa mediante la función movimientos_disponibles que devuelve lo deseado a partir del estado del tablero \texttt{(el atributo "estado")} y el turno del jugador \texttt{(el atributo "turno")}.
    \item \textbf{Movimiento}. Almacena el movimiento \texttt{(representado mediante la tupla: fila, columna)} que ha dado lugar al estado del nodo en cuestión. Este movimiento es siempre nulo en el estado inicial de la partida, pues no se ha realizado ningún movimiento previo a dicho estado. Por ello, su valor por defecto es nulo y debe ser especificado si se quiere construir un Nodo que sí tenga movimiento "creador".

necesito que me escribas la ecuación UCT para poder incluirla en mi documentación .tex

me he dado cuenta de que el uso de \texttt{} con un texto que tenga un guión bajo "_" da error. cómo resolver esto?

y qué hacer con este "warning"?: "Underfull \hbox"

como poner las letras de la leyenda de la ecuación en cursiva?

Cómo añadir sub elementos en una lista en LaTeX

Cómo crear gráficos en LaTeX?

la ruta de la imagen es relativa a la carpeta en la que se encuentra el archivo .tex?

y qué hago para hacer referencia a la etiqueta (label) del gráfico?

Cómo añado enlaces a la bibliografía en LaTeX?

creame una bibliografia MANUAL sin uso de bibtex con los recursos:

S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, 4th ed., Pearson, 2020, ch. 5, "Adversarial Search".

C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Ta
vener, D. Perez, S. Samothrakis, and S. Colton, "A Survey of Monte Carlo Tree Search

Methods,"IEEETransactionsonComputationalIntelligenceandAIinGames ,vol.4,no.1,
pp. 1–43, 2012, doi: https://ieeexplore.ieee.org/document/6145622

estoy utilizando vscode y el pdf se compila de manera automática cada vez que hay un cambio en el .tex

Esos errores se muestran en el texto del propio .tex