Temas Tratados en el Trabajo Práctico 1

- Diferencia entre Inteligencia e Inteligencia Artificial.
- Concepto de omnisciencia, aprendizaje y autonomía.
- Definición de Agente y sus características. Clasificación de Agentes según su estructura.
- Identificación y categorización del Entorno de Trabajo en tabla REAS.
- Caracterización del Entorno de Trabajo.

Anotaciones

"Acordarse de la definición de agente"

Ejercicios Teóricos

1. Defina con sus propias palabras inteligencia natural, inteligencia artificial y agente.

Inteligencia Natural: Es la capacidad de los seres vivos para detectar su entorno, aprender de la experiencia y tomar deciciones. Es una inteligencia que no tiene algo sintetico, no es creada artificialmente. Un ejemplo puede ser un perro detectando un tipo de patron y responder a éste. No es algo presente únicamente en los humanos.

Inteligencia Artificial: Es la capacidad de sistemas creados por el ser humano para imitar ciertas funciones de la inteligencia natural, como razonar, aprender o resolver problemas. Funciona a partir de algoritmos y reglas programadas, y su naturaleza es completamente sintética.

Agente: Puede hacer algo en el mundo. Es una entidad que dispone de algun modo de leer la información útil y realizar cambios en el mundo para realizar su objetivo. En el caso de agentes artificiales, se diseñan para objetivos específicos.

1. ¿Qué es un agente racional?

Un agente racional es, básicamente, un agente que siempre elige la acción que cree que maximizará su rendimiento, según la información que tiene disponible en ese momento y su modelo del entorno.

Percibe su entorno a través de sensores. Decide qué hacer basándose en sus percepciones, conocimientos y objetivos. Actúa buscando obtener el mejor resultado posible según una medida de rendimiento predefinida.

1. ¿Un agente es siempre una computadora?

No necesariamente. El término agente se utiliza para cualquier entidad que percibe su entorno, toma decisiones basadas en esto y actúa sobre su entorno. Por ejemplo, el humano también puede ser considerado un agente racional, cualquier sistema que pueda tomar variables y generar modificaciones en función de esto puede ser considerado un agente. Un ejemplo de un agente que no es una computadora ni tampoco un agente natural puede ser el sistema del apaga velas.Sería un agente simple, que percibe un estímulo y actúa.

1. Defina Omnisciencia, Aprendizaje y Autonomía.

Omniciencia: Capacidad de conocer absolutamente toda la información sobre el estado (Actual, pasado y futuro) del entorno. Ningún agente, ni humano ni artificial, puede ser realmente omniciente. Se puede preparar algo increiblemente completo y cercano a la realidad en contextos limitados, pero no omniciente.

Aprendizaje: Capacidad de un agente de mejorar su rendimiento a lo largo del tiempo, lo que permite que se vayan mejorando los parametros internos en base a la experiencia y a los resultados obtenidos.

Autonomía: Grado en que un agente puede operar y tomar decisiones por sí mismo, sin intervención humana directa. No necesita supervicion para desarrollar correctamente su tarea en cuanto al rendimiento.

1. Defina cada tipo de agente en función de su **estructura** y dé un ejemplo de cada categoría.

Agente reactivo simple

Actúa directamente según la percepción actual, sin considerar el historial ni predecir consecuencias futuras Tienen inteligencia limitada y necesitan que el mundo sea totalmente observable para tomar la decisión correcta
Ejemplo: Una pava eléctrica que se apaga cuando el agua hierve; una puerta automática que se abre al detectar movimiento.

Agente basado en modelos

Incorpora un modelo del entorno para actualizar su estado interno y mejorar la calidad de las acciones.
Ejemplo: Un termostato inteligente que ajusta la calefacción según la temperatura actual y un modelo del tiempo esperado; una cámara que ajusta la iluminación en función del color detectado.

Agente basado en objetivos

Usa el modelo del mundo y un objetivo deseado para decidir las acciones que lo acercarán a ese estado. El agente no se limita a reaccionar a estímulos inmediatos, sino queproyecta el impacto de sus acciones hasta alcanzar el objetivo.
Ejemplo: Un sistema de delivery que planifica rutas para cumplir con la entrega; Google Maps buscando el camino más corto.

Agente basado en utilidad

Elige acciones no solo para alcanzar un objetivo, sino optimizando según una función de utilidad que refleja preferencias.
Ejemplo: Un planificador de rutas que elige el camino más cómodo, evitando semáforos o tráfico, aunque no sea el más corto.

Agente que aprende

Ajusta y mejora su comportamiento con la experiencia, modificando su modelo, objetivos o función de utilidad.
Ejemplo: Un robot aspiradora que recuerda el plano de una casa para limpiar más rápido en cada pasada.

1. Para los siguientes entornos de trabajo indique sus **propiedades**:

    a. Una partida de ajedrez.
        Totalmente observable
        Determinista
        Episodico

Entorno: Estatico
Discreto
Multiagente

b. Un partido de baloncesto.
Parcialmente observable
Estocastico
Secuencial
Entorno: Dinámico
Discreto y continuo
Multiagente

c. El juego Pacman.
Totalmente observable
Estocastico
Secuencial
Entorno: Dinámico
Discreto
Multiagente

d. El truco.
Parcialmente observable
Estocastico
Secuencial
Entorno: Estático
Discreto
Multiagente

e. Las damas.
Totalmente observable
Determinista
Secuencial
Entorno: Estático
Discreto
Multiagente

f. El juego tres en raya.
Totalmente observable

Determinista
Secuencial
Entorno: Estático
Discreto
Multiagente

g. Un jugador de Pokémon Go.
Parcialmente observable
Estocastico
Secuencial
Entorno: Dinámico
Discreto y continuo
Multiagente

h. Un robot explorador autónomo de Marte.
Parcialmente observable
Estocástico
Secuencial
Dinámico
Discreto y contínuo
Monoagente

1. Elabore una tabla REAS para los siguientes entornos de trabajo:

a. Crucigrama.
Rendimiento: Completar todas las palabras en el menor tiempo posible.
Entorno: Tablero con celdas con letras y espacios en blando.
Actuadores: Lápiz o lapicera.
Sensores: Vista, reconocimiento de patrones y letras.

b. Taxi circulando.
Rendimiento: Seguridad, rapidez y minimizar distancias.
Entorno: Calles, peatones, clientes, semáforos, condiciones climáticas.
Actuadores: Volante(dirección), frenos, acelerador, luces, bocina.
Sensores: GPS(o mapa), velocimetro, taxímetro.

c. Robot clasificador de piezas.

```
        Rendimiento: Seleccionar y clasificar piezas
correctamente segun las cualidades seteadas con la mayor
confiabilidad posible.
        Entorno: Cinta transportadora donde se encuentran las
piezas.
        Actuadores: Brazo robótico, pinza de agarre.
        Sensores: Cámara de visión artificial, sensor de color,
detectores de posición.
```

Ejercicios Prácticos

1. La Hormiga de Langton es un agente capaz de modificar el estado de la casilla en la que se encuentra para colorearla o bien de blanco o de negro. Al comenzar, la ubicación de la hormiga es una casilla aleatoria y mira hacia una de las cuatro casillas adyacentes. Si...

- ... la casilla sobre la que está es blanca, cambia el color del cuadrado, gira noventa grados a la derecha y avanza un cuadrado.
- ... la casilla sobre la que está es negra, cambia el color del cuadrado, gira noventa grados a la izquierda y avanza un cuadrado.

  Caracterice el agente con su tabla REAS y las propiedades del entorno para después programarlo en Python:

  ¿Observa que se repite algún patrón? De ser así, ¿a partir de qué iteración?

Hormiga de Langdon

```
Rendimiento: Realizar las instrucciones de la hormiga lo mas rapido
posible.
Entorno: Grilla de cuadros con estado binario (Negro o blanco).
Actuadores: Cambiar de color la celda en la que se encuentra, rotar
90°, desplazarse a una celda adyacente
Sensores: Detectar color de la celda a la cual se desplazó
```

Propiedades:

```
Totalmente observable
Determinista
Secuencial
(Estático en cada paso)
```

Discreto

Monoagente

```
import pygame
import random

GRID_SIZE = 100
CELL_SIZE = 6



WHITE = (255, 255, 255)
BLACK = (0, 0, 0)
ANT_COLOR = (255, 0, 0)

DIRECTIONS = [(0, -1), (1, 0), (0, 1), (-1, 0)]

pygame.init()
screen = pygame.display.set_mode((GRID_SIZE * CELL_SIZE, GRID_SIZE *
        CELL_SIZE))
pygame.display.set_caption("Hormiga de Langton")

grid = [[0 for _ in range(GRID_SIZE)] for _ in range(GRID_SIZE)]

x, y = random.randint(0, GRID_SIZE - 1), random.randint(0, GRID_SIZE -
        1)
dir_idx = random.randint(0, 3)

running = True
clock = pygame.time.Clock()

while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    if grid[y][x] == 0:
        grid[y][x] = 1
        dir_idx = (dir_idx + 1) % 4
    else:
        grid[y][x] = 0
        dir_idx = (dir_idx - 1) % 4
```

```
        dx, dy = DIRECTIONS[dir_idx]
        x = (x + dx) % GRID_SIZE
        y = (y + dy) % GRID_SIZE

        for row in range(GRID_SIZE):
            for col in range(GRID_SIZE):
                color = WHITE if grid[row][col] == 0 else BLACK
                pygame.draw.rect(screen, color, (col * CELL_SIZE, row *
            CELL_SIZE, CELL_SIZE, CELL_SIZE))

        pygame.draw.rect(screen, ANT_COLOR, (x * CELL_SIZE, y * CELL_SIZE,
            CELL_SIZE, CELL_SIZE))

        pygame.display.flip()

pygame.quit()
```

1. El Juego de la Vida de Conway consiste en un tablero donde cada casilla representa una célula, de manera que a cada célula le rodean 8 vecinas. Las células tienen dos estados: están *vivas* o *muertas*. En cada iteración, el estado de todas las células se tiene en cuenta para calcular el estado siguiente en simultáneo de acuerdo a las siguientes acciones:

- Nacer: Si una célula muerta tiene exactamente 3 células vecinas vivas, dicha célula pasa a estar viva.
- Morir: Una célula viva puede morir sobrepoblación cuando tiene más de tres vecinos alrededor o por aislamiento si tiene solo un vecino o ninguno.
- Vivir: una célula se mantiene viva si tiene 2 o 3 vecinos a su alrededor.

  Caracterice el agente con su tabla REAS y las propiedades del entorno para después programarlo en Python:

Tabla REAS:

```
Rendimiento: Como no hay ninguna meta, el rendimiento se puede
considerar por la fiabilidad y la fluidez de la simulación.
Entorno: El entorno es una cuadrícula, de 25 x 25 en este caso, con
celdas que pueden estar tanto vivas como muertas.
Actuadores: Cambio de estado de la celda en cada iteración.
Sensores: Estado actual de la celda (Viva o muerta) y estado de las
celdas limítrofes
```

Propiedades

Totalmente observable

Determinista

Secuencial

(Estático en cada paso)

Discreto

Monoagente

```
import pygame
import numpy as np
import time
import random
#Tener en cuenta que para poder correr el juego es necesario instalar
        la librería pygame y la librería numpy

pygame.init()
height, width = 500, 500

screen = pygame.display.set_mode((height, width))

bg = 25, 25, 25
screen.fill(bg)
nxC, nyC = 25, 25
dimCW = width / nxC
dimCH = height / nyC
gameState = np.zeros((nxC,nyC))
for y in range(0, nxC):
    for x in range(0, nyC):
            gameState[x, y] = random.randint(0,1)



while True:
    newGameState = np.copy(gameState)
    screen.fill(bg)
    time.sleep(0.1)
    for y in range(0, nxC):
        for x in range(0, nyC):
            n_neigh = gameState[(x-1) % nxC, (y-1) % nyC] + \
                    gameState[(x) % nxC, (y-1) % nyC] + \
                    gameState[(x+1) % nxC, (y-1) % nyC] + \
```

```
                    gameState[(x-1) % nxC, (y) % nyC] + \
                    gameState[(x+1) % nxC, (y) % nyC] + \
                    gameState[(x-1) % nxC, (y+1) % nyC] + \
                    gameState[(x) % nxC, (y+1) % nyC] + \
                    gameState[(x+1) % nxC, (y+1) % nyC]
            if gameState[x, y] == 0 and n_neigh ==3:
                newGameState[x,y]=1
            elif gameState[x, y] == 1 and (n_neigh < 2 or n_neigh >
        3):
                newGameState[x, y] = 0
            poly = [((x) * dimCW, y * dimCH),
                    ((x+1) * dimCW, y * dimCH),
                    ((x+1) * dimCW, (y+1) * dimCH),
                    ((x) * dimCW, (y+1) * dimCH)]
            if newGameState[x, y] == 0:
                pygame.draw.polygon(screen, (128, 128, 128), poly, 1)
            else:
                pygame.draw.polygon(screen, (255, 255, 255), poly, 0)
    gameState = np.copy(newGameState)
    pygame.display.flip()
```

Bibliografía

Russell, S. & Norvig, P. (2004) *Inteligencia Artificial: Un Enfoque Moderno.* Pearson Educación S.A. (2a Ed.) Madrid, España

Poole, D. & Mackworth, A. (2023) *Artificial Intelligence: Foundations of Computational Agents*. Cambridge University Press (3a Ed.) Vancouver, Canada