ASSIGNMENT 3

Assignment Date	06 November 2022
Student Name	MAHALAKSHMI J
Student Roll Number	715319106008
Maximum Marks	2 Marks

Introduction

Artificial Intelligence which is also called natural language processing (NLP) and is being used by the branch to automate document processing, analysis, and customer service activities. Three applications include:

1. ***Intelligent document search*** -Finding relevant information from the large number of scanned documents.
2. ***Investment analysis*** - routine analysis of earnings reports and news so that analysts can focus on alpha generation.
3. ***Customer service & insights*** - Using chat box to solve queries of customers faster.

Introduction to Natural Language Processing

Natural Language Processing (NLP) is a branch of Artificial Intelligence enables computers to understand human language and respond in kind. This involves training computers to process speech and text and interpret the meaning of words, sentences, and paragraphs in context.

Human-Computer Interactions

We input speech or text (e.g. typing into a chat box interface or talking to a smart speaker) then the computer converts the text/speech into a format it can understand (e.g. speech to text and words are converted to vectors). This helps computers cluster and classify different words.

Using its own data sets the computer figures out meaning and context. The computer determines an appropriate response and converts it to speech or text that we understand, and responds to us.

```
from easyAI import  id_solve, Human_Player, AI_Player

from easyAI.AI import TT
```
Now, inherit the class from the Two class to handle all operations of the game −
```
class LastCoin_game(TwoPlayersGame):

#Now, define the players and the player who is going to start the game.

 self.players = players

 self.nplayer = 1

#Now, define the number of coins in the game, here we are using 15 coins for the game.

 self.num_coins = 15

#Define the maximum number of coins a player can take in a movie.

 self.max_coins = 4
```
#Now there are some certain things to define as shown in the following code. Define possible moves.
```
 def possible_moves(self):

   return [str(a) for a in range(1, self.max_coins + 1)]

#Define the removal of the coins

 def make_move(self, move):

    self.num_coins -= int(move)

#Define who took the last coin.

 def win_game(self):

    return self.num_coins &lt;= 0

#Define when to stop the game, that is when somebody wins.

 def is_over(self):

   return self.win()
```

```
#Define how to compute the score.

  def score(self):

     return 100 if self.win_game() else 0

#Define the number of coins remaining in the pile.
  def show(self):
    print(self.num_coins, 'coins left in the pile')

if __name__ == "__main__":
   tt = TT()
   LastCoin_game.ttentry = lambda self: self.num_coins
```

Solving the game with the following code block −

```
r, d, m = id_solve(LastCoin_game,
   range(2, 20), win_score=100, tt=tt)
print(r, d, m)
```

Deciding who will start the game

```
game = LastCoin_game([AI_Player(tt), Human_Player()])

game.play()
```

You can find the following **output** and a simple play of this game −

```
d:2, a:0, m:1
d:3, a:0, m:1
d:4, a:0, m:1
d:5, a:0, m:1
d:6, a:100, m:4
1 6 4
15 coins left in the pile
Move #1: player 1 plays 4 :
11 coins left in the pile
Player 2 what do you play ? 2
Move #2: player 2 plays 2 :
9 coins left in the pile
Move #3: player 1 plays 3 :
6 coins left in the pile
Player 2 what do you play ? 1
Move #4: player 2 plays 1 :
5 coins left in the pile
Move #5: player 1 plays 4 :
1 coins left in the pile
Player 2 what do you play ? 1
```

Move #6: player 2 plays 1 :
0 coins left in the pile

2. A Bot to Play Tic Tac Toe

Tic-Tac-Toe is very familiar and one of the most popular games. Let us create this game by using the easyAI library in Python. The following code is the Python code of this game −
Import the packages as shown −

```
from easyAI import TwoPlayersGame, AI_Player, Negamax

from easyAI.Player import Human_Player
```

Inherit the class from the TwoPlayerGame class to handle all operations of the game −

```
class TicTacToe_game(TwoPlayersGame):

   def __init__(self, players):

#Now, define the players and the player who is going to start the game −

 self.players = players

 self.nplayer = 1

#Define the type of board −

 self.board = [0] * 9

#Now there are some certain things to define as follows −

 #Define possible moves

 def possible_moves(self):

    return [x + 1 for x, y in enumerate(self.board) if y == 0]

#Define the move of a player −

 def make_move(self, move):

    self.board[int(move) - 1] = self.nplayer

#To boost AI, define when a player makes a move −

 def umake_move(self, move):
```

```
    self.board[int(move) - 1] = 0
```
Define the loose condition that an opponent has three in a line
```
def condition_for_lose(self):
  possible_combinations = [[1,2,3], [4,5,6], [7,8,9],
    [1,4,7], [2,5,8], [3,6,9], [1,5,9], [3,5,7]]
  return any([all([(self.board[z-1] == self.nopponent)
    for z in combination]) for combination in possible_combinations])
```
Define a check for the finish of the game
```
def is_over(self):

   return (self.possible_moves() == []) or self.condition_for_lose()
```
Show the current position of the players in the game
```
def show(self):
  print('\n'+'\n'.join([' '.join([['.', 'O', 'X'][self.board[3*j + i]]
    for i in range(3)]) for j in range(3)]))
```
Compute the scores.
```
def scoring(self):

   return -100 if self.condition_for_lose() else 0
```
Define the main method to define the algorithm and start the game −
```
if __name__ == "__main__":
  algo = Negamax(7)
  TicTacToe_game([Human_Player(), AI_Player(algo)]).play()
```
You can see the following **output** and a simple play of this game −
```
. . .
. . .
. . .
Player 1 what do you play ? 1
Move #1: player 1 plays 1 :
O . .
. . .
. . .
Move #2: player 2 plays 5 :
O . .
. X .
121
. . .
Player 1 what do you play ? 3
Move #3: player 1 plays 3 :
O . O
. X .
. . .
Move #4: player 2 plays 2 :
```

O X O
. X .
. . .
Player 1 what do you play ? 4
Move #5: player 1 plays 4 :
O X O
O X .
. . .
Move #6: player 2 plays 8 :
O X O
O X .
. X .