Northeastern University
College *of* Engineering

Menace

INFO 6205 Spring 2022 Project

Instructor:

Robin Hillyard

<u>Team Members</u>

Divyesh Darji: 001568627

Shivani Chavan: 001582611

Date: 04/21/2022

Information Systems Department

INFO 6205: Program Structures and Algorithms—Section 01,

Spring 2022

INTRODUCTION:

Professor Donald Michie proposed the Matchbox Educable Noughts and Crosses Engine (MENACE) in the 1960s. This machine is used to automatically learn the Noughts and Crosses game and will grow smarter as it plays against humans. Menace learns to play noughts and crosses by using 304 matchboxes, each loaded with a different coloured bead.

Aim:

To train the MENACE, which is designed and developed by artificial intelligence, to play against human opponents in games of noughts and crosses (tic-tac-toe) by returning a move for any given state of play and to optimize its strategy through reinforcement learning.

Approach:

- The program starts with generating the various states of the game by using a backtracking algorithm for decision making and choosing the optimal move for a player
- These states are used to generate the matchboxes. For each state, exactly one matchbox is generated. Each of these matchboxes symbolizes a different game state
- Each matchbox has a variety of initial fixed numbers of coloured beads *alpha*. Each color corresponds to a different square in the 3*3 matrix
- Because some of the squares have been claimed by markings X, the beads of the excess colors must be removed from the matchboxes
- When the game is over, the machine will check the outcome. If the human player wins, *a gamma* number of beads will be removed from the respective matchboxes which the MENACE used while playing the game against the human. However, if there is just one bead of that color remaining, the bead will not be removed to allow the game to continue smoothly
- If the MENACE wins, the machine will add a beta number of beads to the matchboxes which the MENACE used while playing the game against the human. This increases the likelihood of the MENACE making the same winning move
- After numerous games, the quantity of beads which helps MENACE to win the game in each matchbox will be huge. This indicates that under certain situations, the MENACE has a high possibility of making wise decisions
- If the game ends in a tie, *delta* number of beads will be added to the respective matchboxes which will help MENACE to be trained to draw the match if it is not able to win

PROGRAM:

Data Structures & classes:

The data structures and classes are described as follows:

- **HashMap** is used to store key-value pairs where value is a HashSet that stores all the unique states of the game i.e., the different matchboxes and **ArrayList** to store the number of beads inside the matchbox. ArrayList also stores the current game state of the MENACE which is further used to decide on rewards, punishment and so on.

- **ConsoleContoller.java:**
  This is the main class which is used to run the project. All the training and the console-based game playing activities are defined in this class.

- **Bead.java:**
  This class first fetches the position to place the respective bead for that box in the grid

- **Constants.java:**
  This class defines important constants including *alpha*, *beta*, *gamma* and *beta* which determine the number of beads in a matchbox at every stage in the game

- **GameStatusHelper.java:**
  This class as the name suggests is used to determine the state of the game by passing the current state to determine whether MENACE or human player is the winner, or it is a draw

- **GameStatus.java:**
  This class defines the different status a game can be in, for example, a game is currently running, draw, MENACE wins or human player wins

- **GrandMaster.java:**
  This class is used to train the MENACE in a very optimal manner such that the probability of GrandMaster losing the game is 0

- **Menace.java:**
  This class is used to create matchboxes which represent the states, to reward the MENACE, to punish the MENACE. Also, it contains method to fetch the bead for a particular position

- **MatchBox.java:**
  This class contains method implementations to fill the matchboxes with *alpha* beads at the initial state and then rewarding, punishing and drawing based on the moves played by the MENACE with *alpha*, *beta*, *gamma* and *delta* values

- **StateGenerator.java:**
  This class contains the method to generate the states of the game by using backtracking algorithm

- **SymmetryUtil.java:**
  This class contains the logic to find out the symmetric states for a particular state of a game. We are using different kinds of rotations and reflections to find out symmetric states. Each state has 7 different symmetric combinations

- **LogManager.java:**
  We are implementing log4j as a logging framework for logging each training run with its date and time along with the result of the game i.e. win, loss, draw and probability p. The output is logged in the output.txt file in the src/main/java/output folder

Algorithms:

MENACE Algorithm:

Step 1: Print the Board

Step 2: MENACE finds the first matchbox and plays its turn by picking a random bead

Step 3: Human opponent plays the game

Step 4: Following the human player's turn, this machine will examine the relevant matchbox to determine the next square to be marked

Step 5: Checks the status of the game after each player's turn

Step 6: If the human player wins, *gamma* beads will be removed from each of the utilized beads in the applicable matchboxes

Step 7: If the AI player wins, the machine will add *alpha* beads to each of the utilized matchboxes. This increases the likelihood of the AI player making the same winning move.

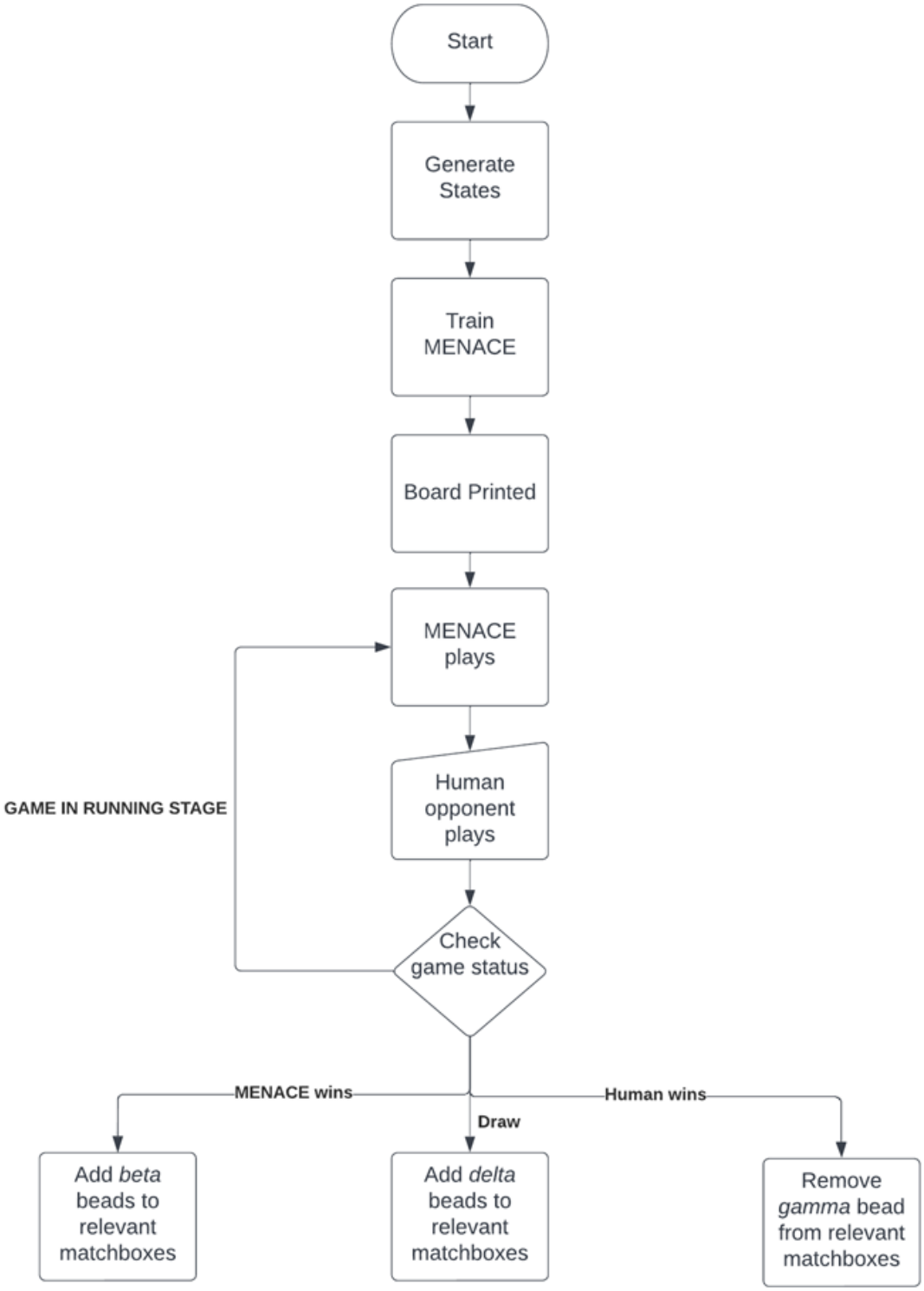
Step 8: If this game ends in a tie, the machine will place *delta* beads in the matchboxes.

Invariants:

The winning positions are the invariant:

- When player wins along rows
- When player wins along columns
- When player wins along diagonal
- When player wins along opposite diagonal

FLOWCHART:


Start
Generate States
Train MENACE
Board Printed
MENACE plays
Human opponent plays
Check game status
GAME IN RUNNING STAGE
MENACE wins
Draw
Human wins
Add beta beads to relevant matchboxes
Add delta beads to relevant matchboxes
Remove gamma bead from relevant matchboxes

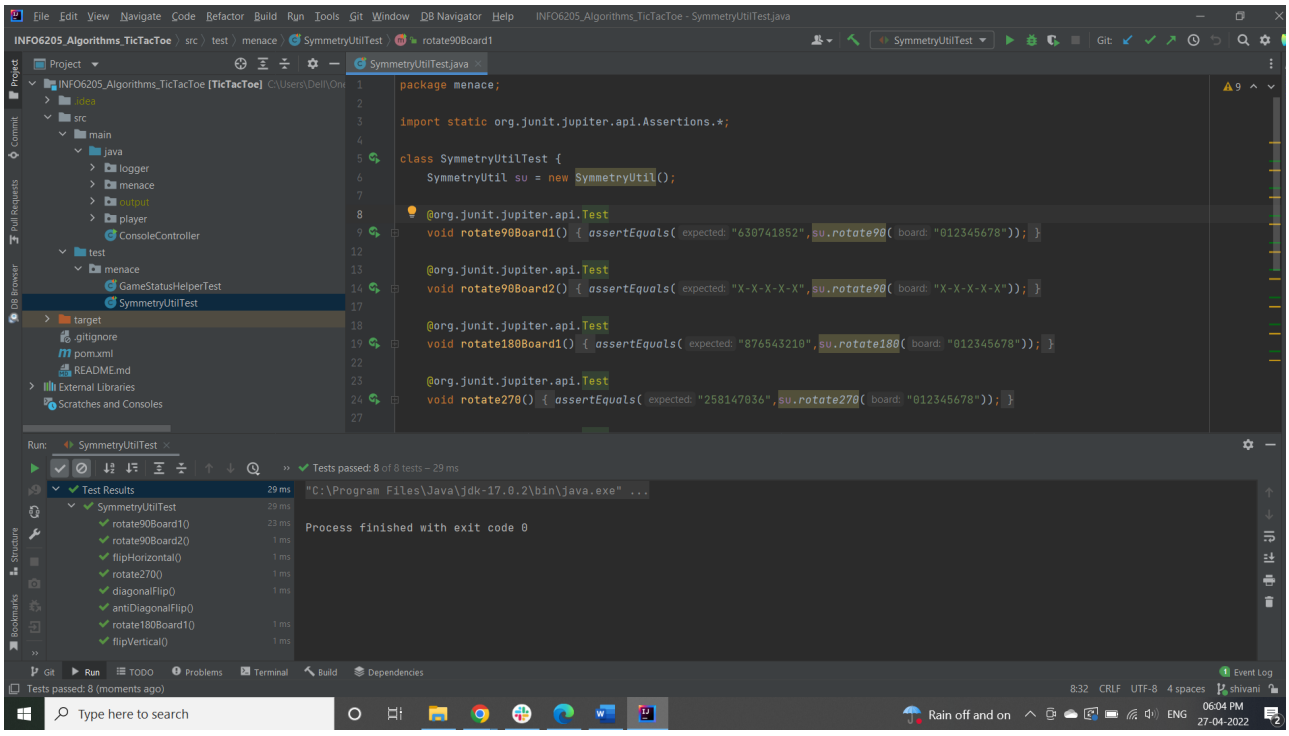
OBSERVATIONS:

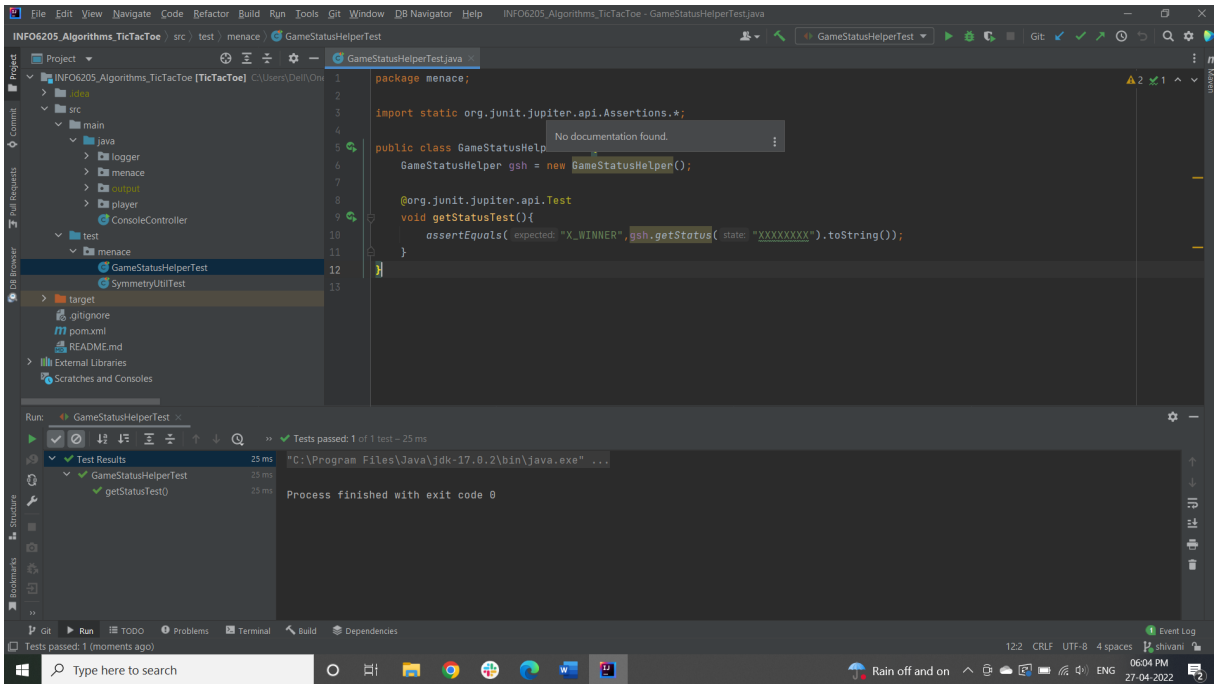
- It has been observed that the when the MENACE was trained for the first time with lesser number of games with the Grandmaster, the moves picked by the MENACE while playing against the human were not very effective to win or draw the game
- This made us to train the MENACE with thousands of games against the Grandmaster so that at the end of the training it will have very a smaller number of beads which makes him lose against a Human in the matchboxes
- We use the concept of reinforcement learning where the MENACE is rewarded every time, he wins the game by adding *beta* number of beads in the matchboxes and punished whenever he loses the game by removing *gamma* number of beads from the matchboxes for every move he played during the game
- As we increase the number of games for training the MENACE it will explore all the new states he comes across and learns what move he should play when it encounters it in the actual game
- Eventually we found that the MENACE started playing very efficiently picking up the beads which will increase the probability of winning and drawing the game
- We are also rewarding and punishing the MENACE while he is playing against the Human so that it can learn while playing the game
- Most of the beads which might lead the MENACE to lose the game are already removed from the matchboxes during the training and thus it is very less likely that the MENACE will lose while playing against the Human
- This is because after the training, there are a greater number of beads in the matchbox which will help the MENACE to select the perfect winning move and the probability of selecting that bead randomly is increased

RESULTS AND ANALYSIS:

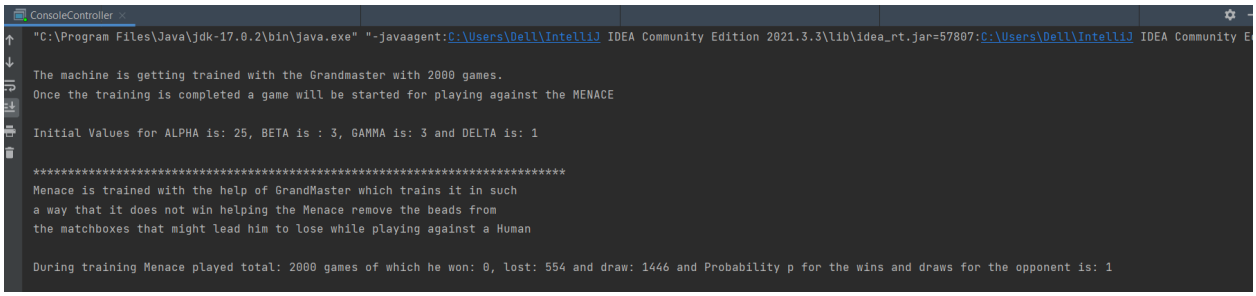
- The probability p for the opponent Grandmaster which is used to train the MENACE at the very initial state is 1 which is a combination of wins and draws out of the total matches played
- This helped us train the MENACE removing *gamma* beads every time it loses against the Grandmaster
- We kept the initial number of beads *alpha* at a value of 25 to train the MENACE and with only 2000 games played during the training. Greater the number of games played during the training, greater will be its accuracy to identify the state and play the same kind of move during the game which he learned during the training
- This helped MENACE learn a little about the various states it could be in while playing the game. The number of beads *gamma* to punish the MENACE plays a very important criteria since it will help us decrease the probability of the wrong bead picked while playing an actual game
- At the end when a human is playing against the MENACE it will make sure to select the moves in such a manner that it won't lose to a human

TEST CASE SCREENSHOTS:





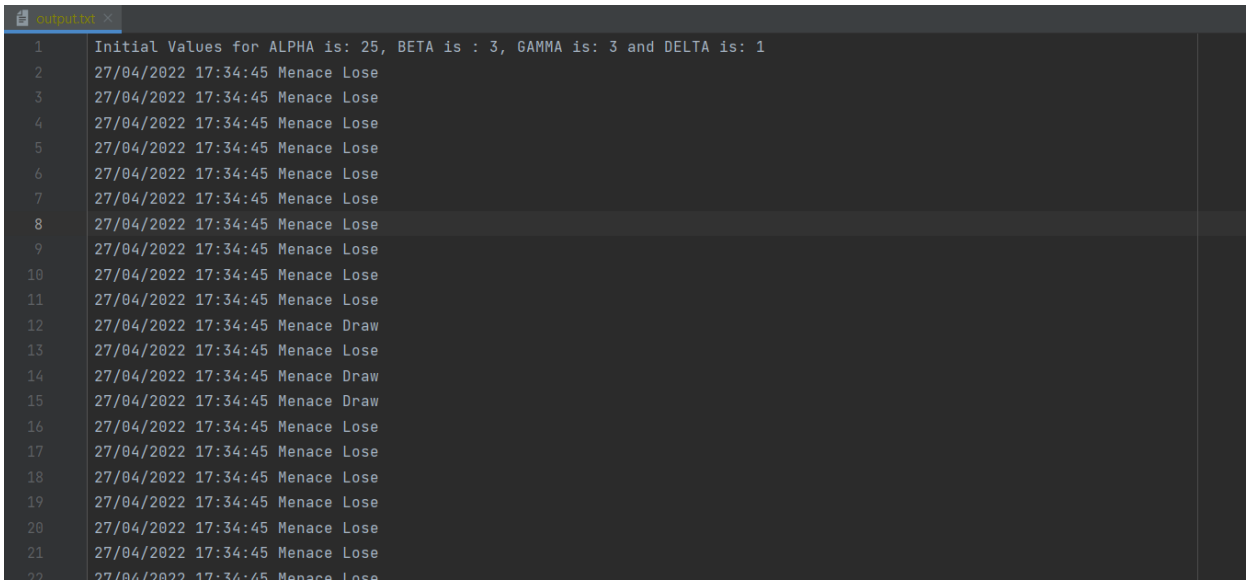
OUTPUT SCREENSHOTS:

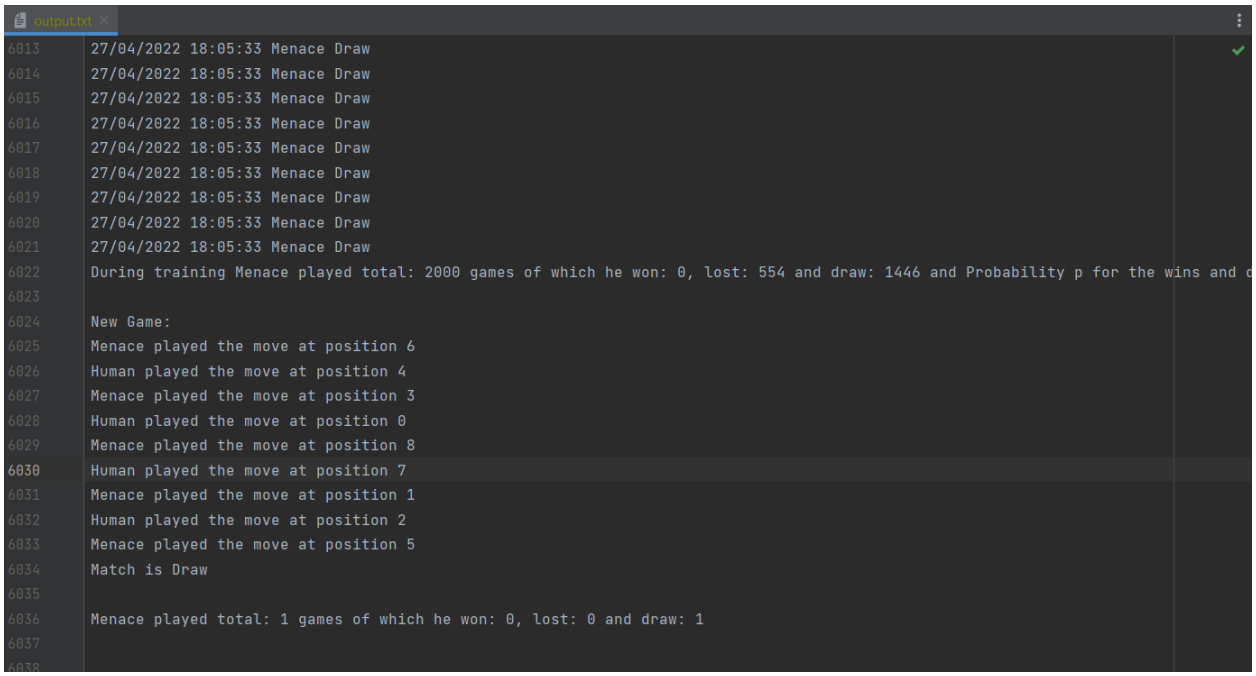


LOGGER FILE:

---

https://drive.google.com/file/d/1IJqbmAGqosuN0QfLxcQ9-zZXBQU1_TYo/view?usp=sharing





CONCLUSION:

MENACE always initiates the first move. On the tic-tac-toe board, MENACE represents 'X'. Each move made by the MENACE and the opponent is represented by different matchboxes. Each matchbox has 9 various colored beads for 9 different positions on a Tic-tac-toe Board. Given that each symmetric or identical move made relates to a matchbox, each victory, loss, and draw in the game has rewards and penalties associated with MENACE by decreasing or increasing beads to the matchboxes. This kind of learning helps MENACE to increase the probability of selecting a random bead from the matchboxes which is more likely to compete against the human while playing the game.