How Can Machine Learning Be Used To Increase Player Understanding

Walker, J.

SID:1503648

MOD002791 Final Project

Final Project Report

BSc (Hons) Computer Gaming Technology

Submitted: April 2018

Abstract

This document will outline the process behind designing an artificial intelligence system that is able to advise players of the game chess, in an attempt to increase their skill levels. This report will cover theories used in the field of AI, heavily focusing on Bayesian statistics and game solving AI. Along with game tree theory and game tree search methods such as the Monte Carlo search method. Brief research into teaching regarding feedback, and skill ranking systems. A design methodology will be given to show how this AI would be implemented using UML diagrams, and discussion regarding the viability of its development within the videogame industry, the effects it can have, and the wider applications of a teaching AI.

Table of Contents

Abstract……..... 1

Table of Contents……..... 2

List of Figures……..... 4

List of Tables…… 5

1.0 Introduction…… 6

1.1 Aims of the study…….. 6

1.1 What is an AI…… 6

1.2 What is Machine Learning …… 7

2.0 Literature Review……....8

2.1 Bayes' Theorem……...8

2.2 Game Tree Theory……...9

2.3 Monte Carlo Tree Search……..……..11

2.4 Video Game AI Scaling ……. 13

2.5 Delivering Useful Feedback……... 14

2.6 Elo Rating System, Calculating Player Skill……..... 16

3.0 Methodology ……18

3.1 Choosing a Language……18

3.2 Designing a Chess Game ……. 18

3.3 Designing Game Trees ……19

3.4 Implementing Datasets …… 22

3.6 Simulating …….. …… 23

3.7 Mimicking Human Play……25

3.8 Feedback Generation.. ……. 26

3.9 Final UML……..…….. 27

4.0 Implementation and Results ……28

4.1 Expected Results for the Player…… 28

4.2 Issues with Implementation……. 28

5.0 Discussion……29

5.1 Wider Applications……..……29

5.2 Limitations of the Design…… 31

5.3 Learning AI as a Mechanic……..…… 31

6.0 Conclusion……..…….. 33

6.1 Overview……..……...33

6.2 Personal Limitations…….. 34

6.3 Further Study…… 35

6.4 Closing Statements…… 35

References……. 37

Appendix I: Poster……..…… 40

Appendix II: Ethical Research Certification……41

List of Figures

Figure 1: UML for a Basic Chess Game……....18

Figure 2: UML for a Game Tree…….…….……..,21

Figure 3: UML for Dataset implementation…….……..22

Figure 4: UML for Calculating Player Skill…….……..24

Figure 5: UML for Feedback generation….…….……..26

Figure 6: Final UML diagram…….…….……..27

List of Tables

Table 1: Proposed Chess Piece Value……12

1.0 Introduction

1.1 **Aims of the study**

In this report I will be researching current methods of machine learning, focusing on self learning. This document will outline the process of building on these methods and designing a process to transfer relevant knowledge to the player. This will be done by assessing the players current skill level, and providing information based on this. The main goal will be to outline a design and implementation methodology to this end. The expected outcome of this, when implemented successfully, will be an overall increase in player skill. I also expect that this form of system will be unable to wholly replace a trained or experienced teacher, but could be used as a teaching supplement.

1.2 **What is an AI**

Tecyci states that "Artificial Intelligence (AI) is the Science and Engineering domain concerned with the theory and practice of developing systems that exhibit the characteristics we associate with intelligence in human behavior, such as perception, natural language processing, problem solving and planning, learning and adaptation, and acting on the environment." (Tecuci 2012). In short an AI is a program that mimics the behaviour of a human being. It is important to define this as there are many misconceptions regarding AI, sci fi often portrays AIs as sapient entities or high functioning versatile computer programs. This is often not the case. It is shown in Video game AI, while not inherently intelligent, often they will mimic Human Intelligence and reactions. This fits Tecyci's definition of an AI.

1.3 **What is Machine Learning**

Murphy defines machine learning as "a set of methods that can automatically detect patterns in data, and then use the uncovered patterns to predict future data, or to perform other kinds of decision making under uncertainty (such as planing how to collect more data)" (*Murphy, K.P. 2012*). Essentially it is a program that analyses data to predict future outcomes (data), Murphy goes on to discuss the three main types of machine learning: predictive (or supervised), descriptive (or unsupervised) and reinforcement learning. Supervised learning is where the correct outputs are known ahead of time, unsupervised learning is the opposite (*Kotsiantis, S.B 2007*). However the type of learning this project will be mainly focusing on is machine learning, this is where the information needed to assess the program's effectiveness is given by an external trainer. For example in a game of chess the program would be able to analyse every move that lead to an outcome (win or lose), and determine which moves where good moves in that situation.

2.0 Literature Review

2.1 **Bayes' Theorem**

Bayes' theorem is a probability equation that takes into account previous known information to reduce uncertainty. For example if there are 200 crates from shipment A and 200 crates from shipment B, the base probability for choosing a crate from shipment A would be 0.5. However if we then declare that 100 crates from shipment A are damaged while only 40 from B are damaged, if we pick out a damaged crate we can show there is a higher chance of that crate being from shipment A.

The Equation is thus

$$P(A|B) = \frac{P(B|A)P(A)}{P(A)P(B|A) + P(not\ A)P(B|not\ A)}$$

Where P is the probability of the event. With A and B being the events.

That is to say the probability of A (Crates being from A) given B (They are broken) is equal to the probability of a crate from A being broken (0.5) times the probability of a crate being from A (0.5). Divided by the probability of A given B times the probability of A, plus times the probability of it not being from A times the probability of it not from A given B.

We can now show that the chance of the broken crate being from shipment A is

$$P(SA|B) = \frac{(0.5)(0.5)}{(0.5)(0.5) + (0.5)(0.2)} = \frac{0.25}{0.35} = 0.714\ or\ 71\%$$

In this case if the crate was not from shipment A, then it had to be from shipment B so the probability of it coming from shipment B 29%.

This is a very simplified version of Bayes' theorem however it is the basis for many algorithms used in computing and statistics. Although Bayesian statistics have been subject to some controversy and is generally considered not viable by older generations of statisions, it makes sense to use it here.

This theory is very effective at increasing the certainty of any probabilities calculated, and is particularly effective when you have a large amount of data about characteristics of the causes of events. In the context of this report the opposing chess AI can use this method during the tree search in combination with a dataset of chess moves in order to determine the human players most probable move. The AI element assisting the player can also use this in conjunction with the all the information gathered about the player to find their most probable move, and asses the outcome to determine it is viable. It can then inform the player of it's findings about this move and if there is a better alternative. However this would require a large amount of data from games that the player has played. In this instance we would use the probability of choosing any certain move, and the probability of choosing a certain move given the players previous choices.

2.2 **Game Tree Theory**

In this section we will be researching methods of AI interaction with the game chess. For this I will be referring to methods of displaying data and sorting the data, these methods will be assuming that it is a two player zero sum game (A game where one player must win, and the other lose).

An important part of this is Game Tree theory. This is where all the possible outcomes (moves) displayed in a tree like structure, each branch (or node) being a game state with end nodes being leafs (*Koller, D. 1994*). As the tree goes on it gets wider but the number of options per node is reduced. There are two types of algorithms for searching a game tree, the first being breadth first: where the each move is considered against moves of the same step as each other. The second: depth first, where the moves are analysed down the branch taking into account all future moves on that path. Although depth first algorithms might be more effective, they are less efficient because they have to see each move through to its ultimate conclusion. A depth first approach is also seen to be a detriment to the quality of the decision (*Nau, D.S., 1983*). While this approach may be necessary against high level players, or against other AI, against the mass majority of players its effectiveness will be indistinguishable from a breadth first approach.

Another approach to representing this data is in the form of a matrix, where each players strategies are labeled on the axis and the success rates (payoff) of the players strategies against the other players are shown inside the matrix. The success rates are relative to player player 1, so if player 2's payoff is larger it will be represented by a negative number (*Barron, E.N., 2013.*). This method is however limited to one decision regarding strategy, and will not be able to change situationally.

In order to rank these moves / nodes a numerical value must be assigned to each move. Many problem solving AIs use a technique called minimax, this is a rule that ensures the minimal maximum loss. Meaning that if there is a loss it should be the lowest amount of loss possible. Meaning in chess every move is analysed with the goal

of prolonging the game while increasing the chance of checkmating your opponent. IBM's Deep Blue chess computer used this method of tree search. One main issue with a minimax approach is that in complex games (such as Chess or Go) it uses a lot of resources to search ahead, as it has to analyse every move possible from that point in the game. A solution to this is to use a Monte Carlo method, that focuses resources on the most viable or interesting states (*Silver, D., 2016*).

2.3 **Monte Carlo Tree Search**

The Monte Carlo method is a loop of 4 states: selection, expansion, simulation and update. Firstly it selects a starting node, then it randomly decides and simulates moves for both itself and the player until the game arrives at a conclusion. If it results in a win, the originally selected node will be increased by a value of 1, otherwise it will result in an increase of 0. It repeats this for every currently available move available at this step, then selects one of the paths with the highest value and the highest number of simulations (ensuring the highest level of accuracy). This is then repeated for the children of this step up until a certain depth updating the values for that path. This can be repeated multiple times at each game state to increase effectiveness. After it does the same with other nodes of the same step. Over time the expansion step will become more weighted towards selecting nodes that haven't been simulated. A way to increase the efficiency of this process further is to examine the goals and constraints of the game, and eliminate potentially unfit selections based on that.

For the example of chess we can focus on the checkmate state, any series of moves that results in a checkmate would receive a 1 for that branch. We could also focus on the value of pieces lost against the value of pieces taken out. To do this we would need to assign a value to each piece, this could be based on the number of said

pieces that are available to the player. Making Pawns the lowest value and the King the highest value. However there many pieces where the same number of pieces are available, to solve this we could then rank pieces with higher mobility at a higher cost (Disregarding the king). This is because they are more versatile and out maneuvering your opponent is key to achieving the checkmate state, and as any piece is able to take another mobility is the most important factor.

Below is a table showing the proposed values:

Piece	Value
Pawn	0.0125
Knight	0.1
Bishop	0.2
Rook	0.3
Queen	0.8
King	1

For example if you were to lose a pawn, but take a queen, the overall gain of that branch is 0.7875. This would more accurately assess a branch's viability. However this approach may be too logical, part of winning a game is predicting what the enemy is going to do. If you are too predictable and your opponent has a high enough understanding of the game they will win no matter what you do. As a result if the AI is too logical the human player may be able to trick the AI into performing a certain set of

actions, which would lead to the human players victory. This would mean that the player would ultimately not learn any useful information about how to defeat another human being in chess, but only become very good at tricking the AI.

2.4 Video Game AI Scaling

While many video games have difficulty settings not many actually tailor the difficulty to the player's skill level. Usually difficulty settings are determined by the player themselves which can lead to them under or overestimate their skill level leading them to be too challenged or not challenged at all. Furthermore at higher skill levels the player will not feel challenged at all, even with the AI difficulty set to the highest setting. One game that does set AI difficulty to match the player's skill is the RTS Star Craft 2, This forces players to beat the easiest of AI before advancing to the more difficult tiers. By winning the player is put up a tier and by losing the player is put down a tier, this ensures that the player is always challenged by their AI opponent. This however does not solve the problem of higher skill players finding the hardest tier trival to beat. One recent development in computer game AI in a competitive setting is Open AI's Dota 2 bot (*Open AI Blog, 2017*). This bot was initially able to defeat professional Dota 2 players, it was created using a self-play system. Where the AI played against itself thousands of times, improving each time with little to no human input. The bot can even implement certain psychological effects into its strategy, such as baiting, where the aim is to goad the player into performing an action to gain an advantage. A similar method could be used to make an AI that can estimate the player's skill level and set itself accordingly, to ensure that the player is always challenged. This could also be taken a step further and attempt to teach the player some of the techniques that it has learned.

2.5 Delivering Useful Feedback

The information gathered by the AI will not be effective if it is meaningless to the player, therefore emphasis on how this information is displayed is necessary. Firstly it is important that the information be delivered to the player at the correct time. “Mowbray (*Mowbray 1953*) provided evidence that, when eye and ear are given complex stimuli at the same time, one or other may give rise to response, but not both.”(*Broadbent D. E. 1956*). Multiple complex stimuli such as assessing an opponent's move in chess, whilst being told complex instructions and reasoning for your next turn. This would mean that the player would filter out one of the sources, leading to a weak or no association between the event and the advice given. Deutch suggests that this “filter” is dependent on the listeners arousal towards the stimuli (*Deutch 1963*). He further shows that words such as the name of the listener rank higher than other words, and will cause the lister to change focus. However this only changes the focus and does not give the listener the ability to interpret both, meaning it is required that the teaching be done after the initial stimuli not during. As a result the most optimum implementation would require a timeout period in between moves before the player is allowed to act, although this will restrict gameplay flow which will make the game less engaging overall. Which is important as no task can be mastered with one attempt (*Ebbinghaus 2013*), the player should enjoy using the application to increase the chance of them returning to it.

An alternative to this would be to show a recap at the end of a game / match, detailing each individual move that was made and what the best response to that move would have been. This may not be the perfect solution as this will be a lot of

information for the player to process in a short period of time. Miller shows that there is an absolute limit on the amount of information that human beings can discriminate within a period of time (*Miller 1956*), any additional information would lead to confusion and would mean that the AI has failed. A potential solution to this problem is to only focus on common mistakes until these mistakes are no longer present (or common). This however will require a large enough samples of data from each of the skill groups before the AI will be able to accurately discern the most common mistakes for each of the groups.

Given the ability to predict a players next move (through Bayesian statistics) a solution might be to pre-warn the player of any impending mistakes that they are probable to make, only notifying the player when they are likely to make a mistake. This would increase the signal to noise ratio, reducing on any unnecessary feedback during the game and therefore reducing the possibility of confusion occuring.

As the AI will primarily targeting experiential learners (due to the nature of the player acting in the game's environment) we can use Kolb and Fry's experiential learning model (*Fry, R. and Kolb, D. 1979*). Although the learning model is cyclical it makes most sense to start from the Concrete experience phase, this is where the student initially attempts the task. It's highly likely that they will fail at this stage, and in fact it is better if they do, the important part is that they have some information as to how they have failed for the next section of the model. After the initial task the student is given time / resources to focus on observations and reflections (Observations and reflections phase). Then conceptualise ideas into theories (Abstract Concepts and Generalisation

phase). Finally they then use these theories to solve problems with their performance (Implications of Concepts in New Situations phase) and starting the task again generating a new set of results. To apply this to chess would be this: firstly the player plays a game of chess against an opponent, they lose or win, reflect on the weaknesses in their strategy, think of how to overcome these weaknesses, implement them into their new strategy and play against another opponent. It is now clear that the most opportune moment for the AI to offer advice is after the player has had a chance to review the match during their conceptualisation of theories. However it would aid the players memory of events if the AI could provide an easily viewable and understandable recap of the players moves to aid in the observation phase of the cycle. How the player would implement this would be their own responsibility as if the AI plainly states this, the player would not learn anything.

2.6 **Elo rating system, Calculating player skill**

In order to provide useful advice and to tailor the AI to the players current skill, firstly their skill needs to be measured. This is commonly done in most zero sum games using the Elo rating system. The Elo system was created by Arpad Elo, and designed for rating chess players. It is a system where a numerical representation of the players skill is assigned an amount is deducted or added to this number on a win or a loss, this amount is decided based on the difference in the players skill (*Elo, A.E., 1978*). The equation is this:

$$R' = R + K(S - E)$$

Where R' is the new rating, R is the old rating K is the maximum increase or decrease of rating (for chess this is 32 unless they are the rank of master (2000+) which in which

K is 16), S the actual outcome and E the expected outcome the expected outcome of a player is calculated as so:

$$E_A = \frac{1}{1 + 10^{(R_B - R_A)/400}}$$

Where E is the expected outcome of player. A and B being the two players, and R the rating of the player. The equation for player B would look like this

$$E_B = \frac{1}{1 + 10^{(R_A - R_B)/400}}$$

This is used as it provided diminishing results for beating players of a lower skill meaning a player cannot reach higher ratings by just beating low skill opponents.

For example if player A's Elo rating is 1500 and player B's rating is 1000 the expected outcome for player A would be this:

$$E = \frac{1}{1 + 10^{(1000 - 1500)/400}} = \frac{1}{1 + 10^{-1.25}} = \frac{1}{1.056} = 0.94$$

meaning if player 1 won their Elo rating would be

$$R' = 1500 + 32\,(1 - 0.94) = 1500 + 1.92 = 1501.92 = 1502$$

If player 1 lost their Elo rating would be

$$R' = 1500 + 32\,(0 - 0.91) = 1500 + -29.12 = 1470.88 = 1471$$

Elo rating generally starts at 1000 and new players are often given a higher K value so they can rise or fall through the rankings during there provisional games more quickly, and their opponents will often lose less Elo as it is highly likely they will not be at an accurate rank. during these games.

3.0 Methodology

3.1 Choosing a language

The language chosen for this project is C++ this is because a lot of flexibility is provided by the use of pointers and allowing for explicit memory allocation, along with this C++ is more efficient which is important seeing as large amounts of data needs to be processed.

3.2 Designing a simple Chess game

As the main focus of the report is not the design and implementation of the chess game itself, the design has been oversimplified. Although all the necessary elements are there. There is a board, which contains a list of spaces and methods to set up and display the board. Each space has a position and a pointer to a piece that occupies that space. Finally we have the main game loop where most of the logic for the game will take place.

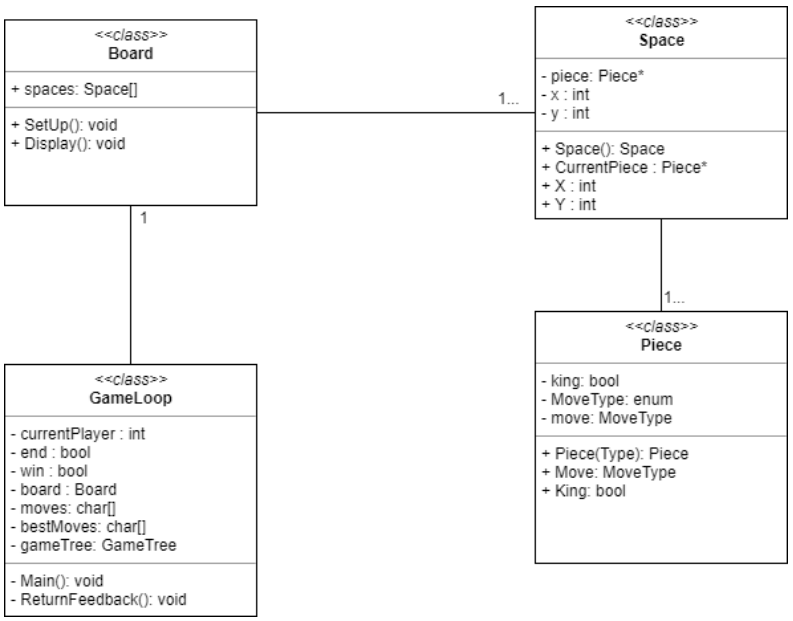


Fig. 1 UML for basic chess game

Each piece will have a move type variable that will contain the direction that that piece can move in.

3.3 Designing Game Trees

The core part of the AI will be the concept of Game Tree Theory. To implement this a form of nested list will be used. In C++ there is no type list as there is in C# which means that an object must be created to store multiple objects. However if C# was used it is likely that a new list class would have to be created to mimic the tree like structure. Firstly individual nodes need to be created (Listed as TreeNode on the UML diagram).

Nodes should contain pointers to the nodes to the left, right, above and below (all children), this is so the program knows which node to go to next when iterating through the list. Although it is not necessary for the Monte Carlo method as a child node will be selected from the parent node without checking against other nodes, if at anypoint another search algorithm is used the ability to compare nodes of the same hierarchy would be useful. Nodes will also need a string to represent the move taken these will be stored in Standard Chess Notation (SCN), If a bishop has moved from d4 to e5 it will be noted as Be5 the B representing the piece (Pawns movement uses no prefix). This is important as the dataset that will be used will be in this standard notation. Each node will also contain how many times that particular node has been simulated (used for monte carlo search as previously discussed). Each of these variables will have a public accessor used for comparing them.

A GameBranch functions similarly to nodes in the respect that it has pointers to the previous and next GameBranch but they also contain pointers to the first and last Nodes within that branch, it also has a pointer to the current node (used for iterating through nodes). Each branch will also have an integer variable used to store the total cost of that branch, along with a bool variable to store whether or not that branch is viable. A Branch will have public accessors to both the next and previous along with two other public methods. The 1st being PopulateList, which will add each of the possible moves from the start of the branch until the that branch's conclusion. The 2nd a method called EvaluateBranch which will be responsible for totaling the cost of the branch and, if at the end it becomes less viable than a previous branch, change the viable bool accordingly. This method will be given this lowest cost variable from the GameTree class.

The game tree class will also have start, end and current variables except they will be pointers to GameBranchs rather than TreeNodes. It will also contain the lowestCost integer and a pointer to the GameBranch associated with this. This class also contains a PopulateList method that will be responsible for generating each GameBranch and having it run it's own PopulateList method. Finally there is a search method that will iterate through each branch's EvaluateBranch method and update the lowest cost variable if needed. These methods will be called from within Main() inside the main game loop when it is the AI's turn.

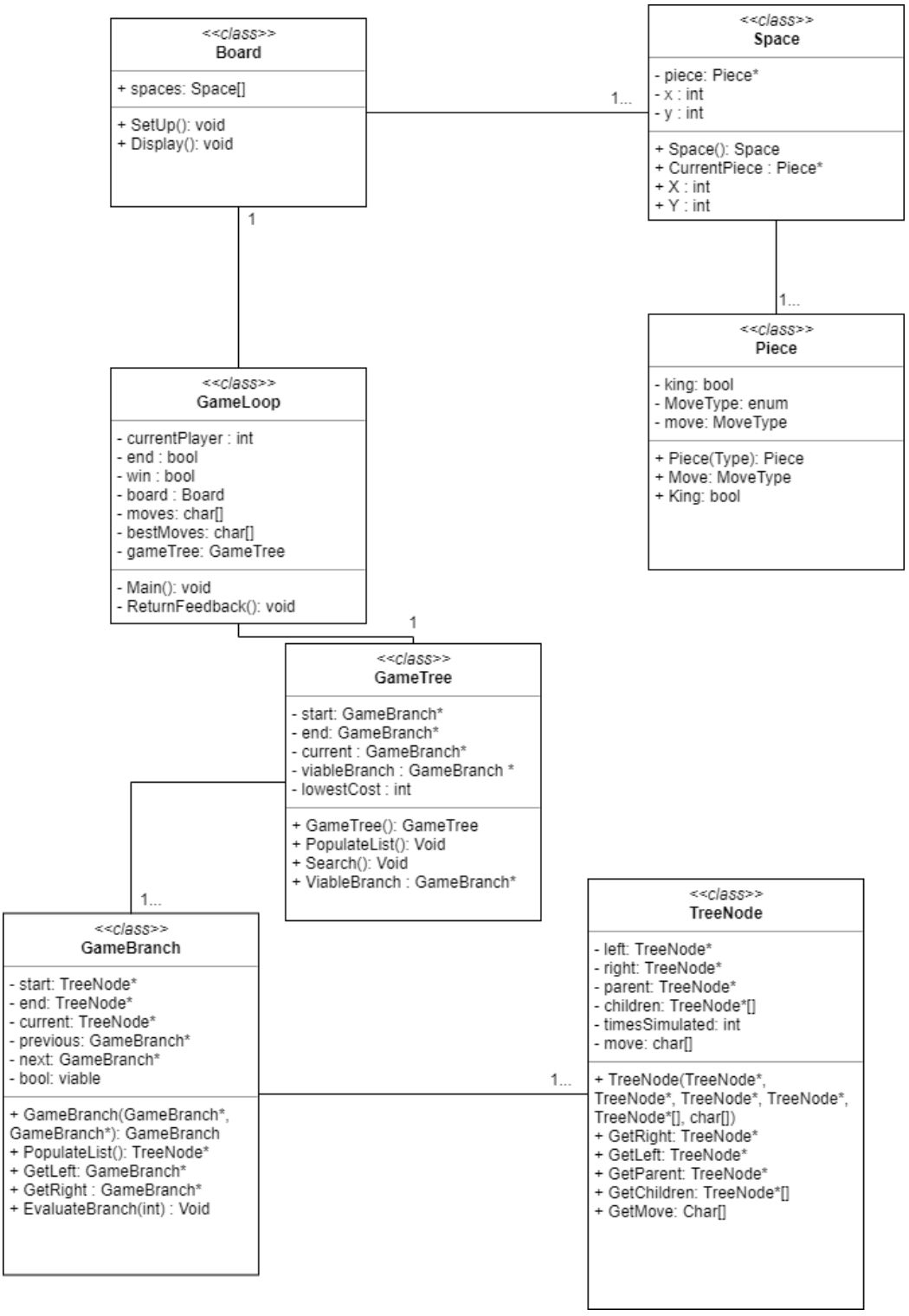


Fig. 2 UML for a Game Tree

3.4 Implementing Datasets

In order to simulate the players moves we first need to implement a class to handle the loading of a dataset so it can be analysed. Given enough time the AI could generate it's own, however at this point it would be simpler to use a given set. This will have a more varied set of moves as the data will not all be from the same player. The dataset provided with this document (*Kaggle 2017*) contains the information required to predict the players move, data such as: winner, moves taken (in SCN), player skill rating and opening strategies.

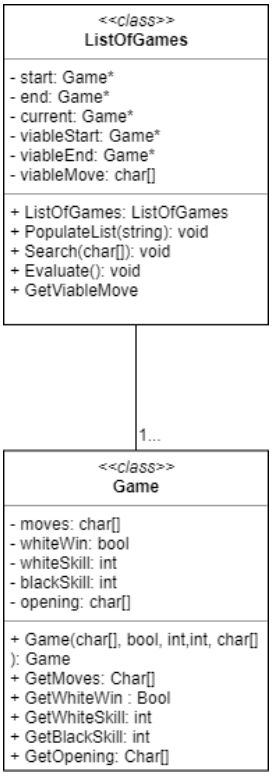


Fig. 3 UML for Dataset implementation

These classes are set up in a similar way to the GameBranch and GameNode classes. The node in this instance is the Game class, this class contains all the information for the AI to make an informed decision when it comes to predicting the players next move. It has variables for the previously mentioned data, along with public accessors to each.

The list of games class is different from the lists designed before as it has two variables for the start and end of a list, this is because 2 lists are needed. One for all the games in the dataset, and the other is for games that are specific to the current state of the game. This class also contains a char array that will store the most likely move the player will make.

List of games has a populate list method that is similar to the other lists in the project other than it will be loading data from a Comma Separated Values (CSV) file using the fstream library. After the variables are loaded from the CSV file a new Game object will be created using it's constructor and inserted into the list, this list will only be populated when a ListOfGames object is created. Once the list is populated the Search method can be used, this method takes a char array of all the moves up until this point and populates a list of Games that matches these moves. Although the dataset provided is quite large, there are around 72,000 possible combinations after 2 moves, so this will need to search for similar configurations not exact matches. Once a set of probable moves are selected, the evaluate method will be run. The Evaluate method implements bayesian statistics researched earlier.

3.5 Simulating

Once the data is loaded the move can be predicted, this can be done by implementing bayes theorem. First select a viable move which the player is in the position to make, then we take the base probability of that move (based on the amount that move is made

out the total moves that can be made). Then take the probability of the player taking that move based on their skill rating, when put into the formular it will look like this:

$$P(A|B) = \frac{P(move\ given\ skill)P(move)}{P(move)P(move\ given\ skill) + P(other\ move)P(other\ move\ given\ skill)}$$

This calculation will take place in the Evaluate method inside the ListOfGames class, each potential move will undergo this process if the move has a higher chance than the current viableMove variable this move will replace it. Once each move has been evaluated the viable move is used to update the game state for the next move.

3.6 Calculating player skill

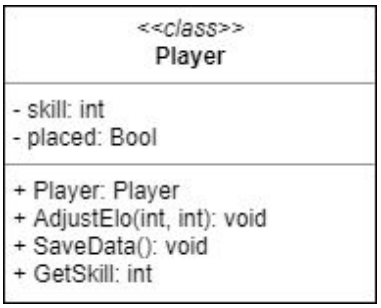


Fig. 4 UML for Calculating Player Skill

To determine the level at which the AI will play at and what advice will be given, a player skill rating is required. This is shown as an Elo ranking, to attain a rank the player must undertake a set of 20 placement games. During these placement games the AI will be required to increment in difficulty in order to accurately place the player. To calculate and store this value the Player class is created. This class will contain variables for the players current skill rating (Starts at 1000) and how many games the player has completed. Along with this the class will contain 3 methods and a constructor. The first method will be responsible for adjusting the players Elo rating, this will take 2 int variables the opponents Elo rating and the outcome of the match (0 if a loss, 1 if a win). This method will use the calculations discussed in chapter 1.9 to

calculate the players new Elo rating. The next method will be used to save the players data externally so it can be loaded on different executions of the program, this number will be stored along with other information about the player this could include game data (e.g. moves made, outcome, skill level of the game). When this information is being loaded it would be based on the most recent game data. Finally there is a public accessor for the players Elo ranking that will be used by the AI to tailor it's play level.

Adjusting the AIs Elo rating simply changes the range of moves that it will consider based on moves from the dataset.

3.7 Mimicking Human Play

Now that the players skill has been decided the AI can now tailor it's moves to that skill level, ideally the AI would play at a slightly higher skill so the player learns more from the experience. Where previously the AI would search for the best move and then perform it, it will now search for the most used move by human players in that situation. The methods that would search for the best move will now be given to the player as feedback on moves that they have made.

This method would work in a similar way to the prediction stage of simulation where the AI will take the most probable move of a player of that skill given the player's predicted next move. This should help the player more in actual player vs player situations as the AI would be essentially copying the most used move in that situation. For masters and above ranked opponents it could be more beneficial to allow the AI to find the best move as it could provide a better learning experience.

Functionally the core methods will not change, the ListOfGames methods will be ran twice the 1st being the players move (Predictive) the 2nd being the AI's next move. The GameTree's viable move will be passed to the player via feedback instead informing the AI's next move.

3.8 Feedback Generation

Generating feedback for the player is a relatively simple task, to do this all moves of the game should be recorded along with all the best possible moves. At the end of a game these sets of moves will then be displayed after a game. This will be done by adding two new char array variables to the GameLoop class to store these sets, and adding method that will display them. Seeing as the aim is to create a console application these will be output to the console, however a more intuitive GUI could be employed showing a step by step of moves all the moves they made and which moves would have been better. The challenge comes from generating feedback that is accessible to new players but expert players will still find useful.

Fig. 5 UML for Feedback Generation

3.9 Final UML

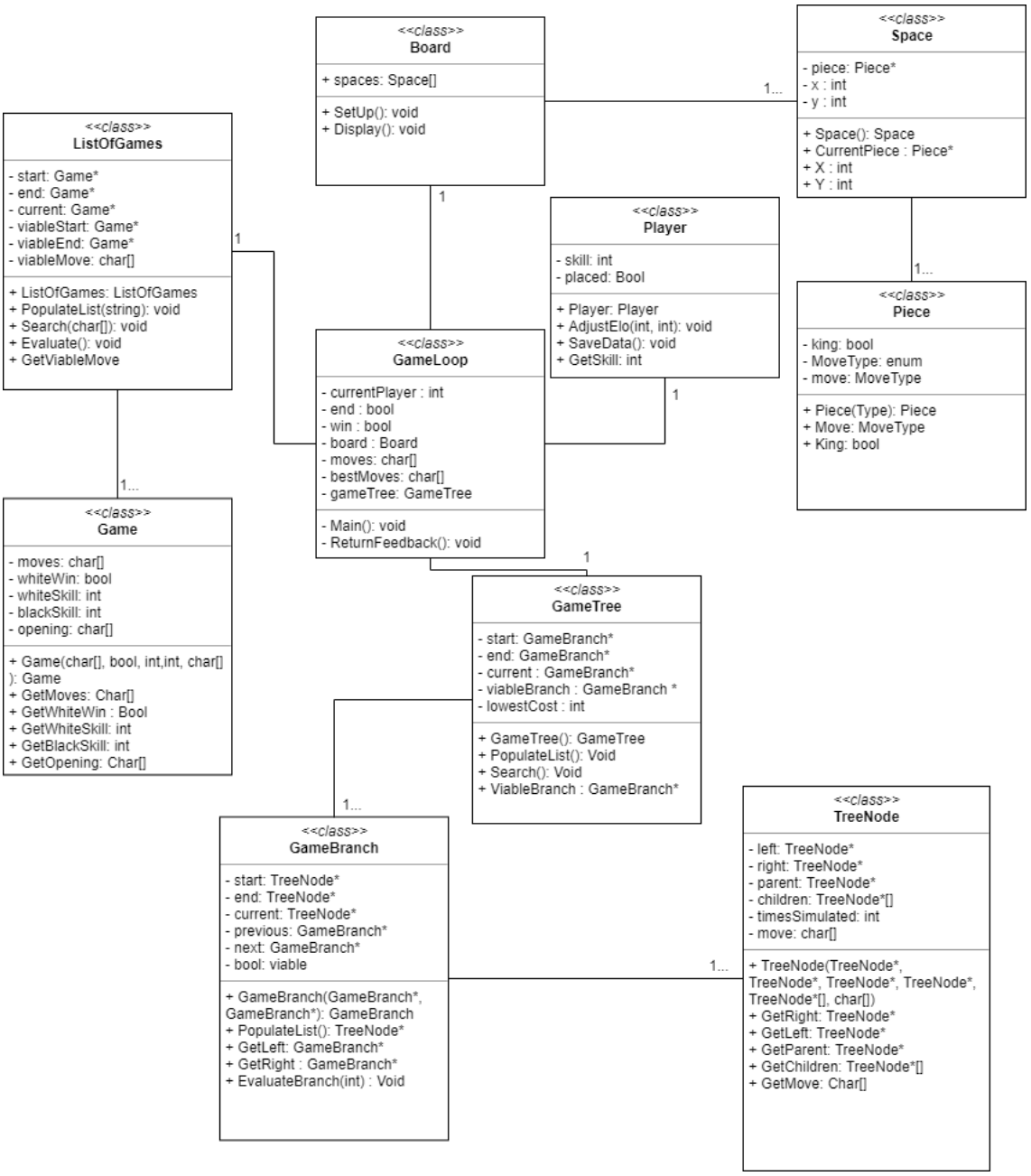


Fig. 6 Final UML

4.0 Implementation and Results

4.1 Expected Results for the Player

What I would expect to see from exposing people to this type of AI is a slight increase in overall skill (Elo rating) in players that use the application, compared to those who do not. However it is expected that completely new players would not benefit from these types of applications because there is too much terminology and too many rules that they would have to know before they can start improving upon their skills. A basic understanding of the game would be required before the player would be able to understand why one move is better than another. Furthermore it is not expected that there will be a large change in skill (as opposed to playing the game without assistance) this is because players will get used to being told how to improve, and when they are not able to be explicitly told they will find it difficult to understand their shortcomings.

4.2 Issues with implementation

The goal at the start of this project was to design and implement a learning AI that could offer chess players insight into their mistake, although the aim was to research the topic in a more general sense and only to use chess as an example. Unfortunately due to personal oversight in aspects of the project, it was unable to be implemented. This was a result of a multitude of issues, chiefly among which was a lack of current academic experience. As a result only a design methodology was produced. This will be further explained within the Conclusion of the project.

5.0 Discussion

5.1 Wider Applications

It is apparent based on the literature review chapter that the application of statistics and probability in AI is very prevalent especially Bayesian statistics. Due to the nature of this this approach to AI, and it's ability to reduce uncertainty, has much wider applications. Any task that has a fail state and a win state can utilise this method, it doesn't necessarily have to be zero sum but so long as there are clear success and failure conditions this method is applicable. Meaning it can be utilized to teach subjects other than games. The core concept simply compares the user input with a objective AI generated input and outputs the difference between the two. While there are services that already offer the same end result, having the program inform a student whether or not an answer is correct These programs are explicitly given the answer to a proposed question, Whereas the design of this program would allow it to determine the most efficient answer without having to be told.

The concept of skill rankings applies to anything that isn't a subjective subject (such as art), and Elo is widely used in video games or at least heavily inspires ranking systems. Theoretically you could rank students based on scores or the time taken to perform a task and compare them to each other individually, and calculate an Elo rating based on their performance out of that set of students. This could then be used to determine who should attend higher classes and who should attend the lower classes. This however would not work with subjects such as art for two reasons, firstly there is

not a divide in art classes based on skill. Secondly the measure of success of a piece of art is subjective and has no right or wrong answer (No win / loss state). While it may not be necessarily be a good idea to introduce a system that grades students based on eachothers merritts, there are similar systems in place for students and in most cases schools are ranked against each other.

This kind of AI could be used in the games industry, which recently has been favouring competitive elements. This has caused an influx of high skill cap mechanics in games, which causes a barrier to entry. This result is one that game developers strive to reduce, as their goal is to make their game as accessible or enjoyable as it can be (in most cases). The implementation of a teaching AI inside the game itself would help players, that are disinterested due to this barrier to entry, to quickly come to grasps with the core mechanics and get to a level of skill where they can enjoy the game. This would also interest players of any skill, as due to the competitive nature of these games there is a constant want to improve from the player's perspective. There are actually already available coaching services where another person listens in and watches people play video games and tells them how they can improve. This is as a result of the recent popularisation of esports (Electronic sports), and whole careers that have been made off of this trend. I game I previously mentioned in this report (Dota 2) has recently added a paid analytics service (Dota Plus) that offers the player insight into the game. It will give them suggestions based on the current match they are playing, for example which Hero (Character) to pick in order to counter an enemy Hero or lineup and even which skills to level based on that. It also gives the player post game analytics and trends.

There is currently no data available for how many subscribers there are to this service, however the fact that it exists alone shows there is a demand for it.

5.2 Limitations of the Design

A limitation of this design is that although the AI can arrive at it's own conclusions it still requires programming with specialist knowledge in mind. using this design as an example, if the aim was to create a chess AI it would have to be designed with the rules of chess in mind. It is very inflexible and you cannot simply transfer it to any game and have it learn it from scratch. This could be rectified by implementing natural language processing, like IBM's Doctor Watson. Doctor Watson is a teaching assistant (question answering AI) that was originally created to solve the game Jeopardy (Based on the TV show) but has since gone on to be used as a chatterbot to converse about childrens toys. This AI is able to interpret language and produce an output but cannot comprehend the semantics. Implementing this methodology would allow the AI to interpret rules, but would not be able to understand the meaning behind them. They would just be viewed as lines and symbols.

5.3 Learning AI as a Mechanic

Previously discussed was the AI's potential for a competitive gaming environment, however learning AI is already used as a mechanic in games. In games such as Hello Neighbour, a game where you have to break into your neighbour's house in order to gain knowledge of their sinister deeds. The neighbour (AI element) in question will learn things about the way you play the game, in an attempt to stop the player from achieving their goal. For instance if you (the player) constantly enter the house through a window, a trap would be placed neer that window. The difficulty is increased each

time the player is caught by the neighbour. This is used to give the impression of an intelligent opponent and to make the game more interesting, and increasing difficult, the more times you play it.

This style of mechanic could be interesting when implemented into strategy games. Take the game sid meier's civilization, in the game a player controls a nation's leader with the ultimate goal of one of the many win conditions, being no other players remaining, technological advancement, culture and diplomacy. The game features AI advisors which inform the players on what they should build or research. Implementing a teaching AI into this mechanic would lead to situational results based on how the player wants to interact with the game. This would mimic the effects of a real human advisor, and could even assume different personalities that give different styles of advice making them seem more human. Variants of this mechanic could be implemented into other genres, for example this level of adaptable gameplay would lend itself to the horror genre to keep players on edge.

6.0 Conclusion

6.1 Overview

The aim of this project was to increase player understanding via the use of an AI element. Although I had managed to research elements of AI development and designed an AI to this end, ultimately I was unable to implement this into a working AI. As a result I was unable to generate a set of data to prove that it is viable, or that it isn't viable. However it is possible to look at current AI that attempt to perform this task, the previously mentioned teaching aid Dr Watson (Enlight / Element) has programs to aid in early and later stages of education by targeting the students strengths and weaknesses. Although IBM's AI is much more complex than the one I have detailed in this document, the core principle is the same: understand the mistakes that a student could make, and offer this information as feedback. We can see by the continued investment into these technologies from IBM, that this style of system is viable in education and by extension in chess tutoring.

Overall the design of the program outlined is somewhat oversimplified, It portrays a broad interpretation of how such a system could be implemented and how to apply data to actions made. However there is a lot of parsing of data which could cause performance issues, fixes to which have not been mentioned in the methodology. In fact there is very little in the way of optimisation mentioned during the mentioned section. The design currently imports a dataset rather than parsing the information in a file, this means that more data is having to be stored at runtime. While the dataset mentioned is

relatively small (In terms of datasets) having all of it loaded into memory is inefficient. Furthermore the handling of storing new data isn't gone into in detail. However the design is unable to go into great detail due to an inability to foresee potential issues that would arise through the process of attempting to implement it in code. As in any projects, a high level of planning does not always ensure a guaranteed success and there are often unexpected errors or bugs.

6.2 Personal Limitations

A lot of issues I would have in implementing this design effectively is due to my current academic level, and lack of specialist knowledge required to produce an AI in the field of teaching. For instance Chess is a complex game with many nuanced rules such as castling, where in certain situations the king and rook can essentially switch places with each other. Small rules like this require a working knowledge of the game to implement, along with testing the AI. In order to test if the AI was giving the user the best move, knowledge of what the best move should be in some situations is required. Along with this is the lack of knowledge in the subject area of teaching meant that even if I could develop this AI it would most likely be inefficient when it comes to actually teaching players the game. Seeing as there are qualifications required before anyone can start teaching, any attempt I would make would not meet these standards. Finally the standard of programming experience required for this project is higher than my own, the time it would take to implement I would have most likely gone past the deadline that was set. This project is more suited to a Computer Scientist or an AI focused student.

6.3 Further Study

In order to properly implement this design along with the proposed addition of Language processing and adaptability would take further study in this field, if it were to be taken forward to a higher level it could be successful. And data could be gathered to determine if having an AI to assist you in tasks will increase the long term skill of a user or if it will just show a temporary increase. Taking this further would entail adding this system to other games in an attempt to make a more versatile AI and to gather more data on different subjects. It would also be beneficial, if the project was taken far enough, to attempt to implement this system in a learning environment to understand how students interact with it in comparison to the regular curriculum. The main issue with further development of this is that some specialist knowledge in each of the subject areas expanded into would be required. Consultants would be needed to implement and test each field of study to ensure they are correct, and relevant information is given at the right time.

6.3 Closing Statements

To conclude the research and design element of the project went as planned, although the implementation and results sections didn't. The design as a whole was a success but the aims of the study were not met, and no data was received to prove that the AI would be effective. Further study would be required to fully finish this project, however the project would likely benefit from other researchers of different backgrounds. Although it could be beneficial to the field of game development, it is most likely better researched and used by computer scientists and in some cases psychologists. Focused

more towards academic pursuits, rather than video game implementation. The question is: will this AI be more, less or the same level of use to a user than a set of static instructions? an answer impossible to know without statistics from tests.

References

Barron, E.N., 2013. Game theory an introduction 2nd ed., Hoboken, N.J.: John Wiley & Sons, Inc.

Broadbent, D.E., 1956. Successive responses to simultaneous stimuli. Quarterly Journal of Experimental Psychology, 8(4), pp.145-152.

Deutsch, J & Deutsch, D, 1963. Attention: Some Theoretical Considerations. Psychological Review, 70(1), pp.80–90.

Ebbinghaus, H., 2013. Memory: A contribution to experimental psychology. Annals of neurosciences, 20(4), p.155.

Elnaggar, A.A., Abdel, M., Gadallah, M. and El-Deeb, H., 2014. A comparative study of game tree searching methods. Int. J. Adv. Comput. Sci. Appl., 5(5), pp.68-77.

Elo, A.E., 1978. The rating of chessplayers, past and present. Arco Pub..

Fry, R. and Kolb, D., 1979. Experiential learning theory and learning experiences in liberal arts education. New directions for experiential learning, 6, p.79.

Mitchel. J, 2017. Chess Game Dataset (Lichless) | Kaggle [online] Available Through: https://www.kaggle.com/datasnaek/chess

[Accessed 17/04/2018]

Koller, D., Megiddo, N. and Von Stengel, B., 1994, May. Fast algorithms for finding randomized strategies in game trees. In Proceedings of the twenty-sixth annual ACM symposium on Theory of computing (pp. 750-759). ACM.

Kotsiantis, S.B., Zaharakis, I. and Pintelas, P., 2007. Supervised machine learning: A review of classification techniques.

Miller, George A., 1994. The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. Psychological Review, 101(2), pp.343–52.

Murphy, K.P., 2012. Machine learning a probabilistic perspective, Cambridge, MA: MIT Press.

Nau, D.S., 1983. Pathology on game trees revisited, and an alternative to minimaxing. Artificial intelligence, 21(1-2), pp.221-244.

Open AI Blog, 2017. More on Dota 2. [online] available at:
<https://blog.openai.com/more-on-dota-2/>
[Accessed 20/01/2018]

Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M. and Dieleman, S.,

2016. Mastering the game of Go with deep neural networks and tree search. nature, 529(7587), pp.484-489.

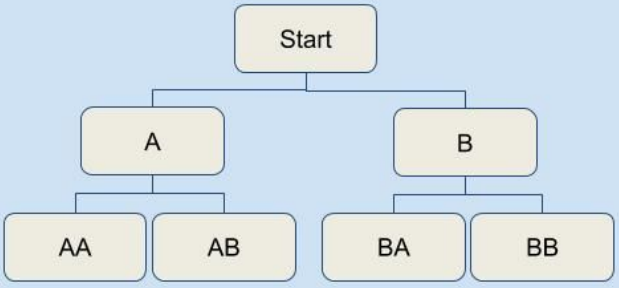
Tecuci, G., 2012. Artificial intelligence. Wiley Interdisciplinary Reviews: Computational Statistics, 4(2), pp.168–180.

Joshua Walker **SID: 1503648** **Computer Gaming Technology**

Using Machine Learning to Increase Player Understanding

Game Tree theory

This is where all the possible outcomes (moves) displayed in a tree like structure, each branch (or node) being a game state with end nodes being leafs (Koller, D. 1994). Search algorithms are used in conjunction with a tree to find the best move.



Adjusting to player skill

By adding another layer to this kind of self learning process, after gauging a players skill level, the AI could give relevant feedback based on their most probable action. This would hopefully improve player understanding of the game.

Other Applications

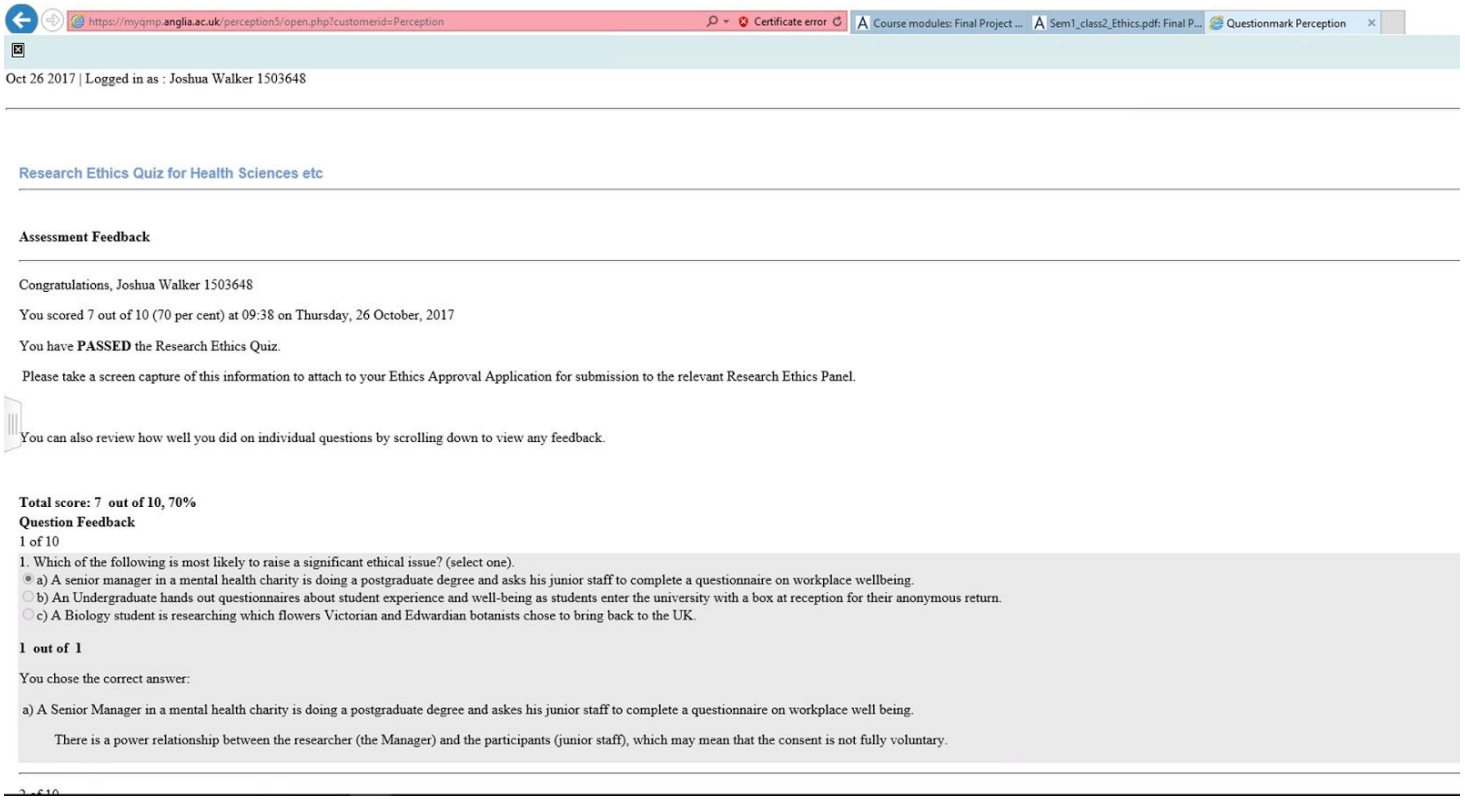
If implemented properly this method could be uses in any field of study, and could be used to teach the player practical skills through the use of serious games.

Open AIs Bot

In 2017, Open AI released their self learning bot for Dota 2. This AI used

Appendix I: Poster

Appendix II: Ethical Research Certification



https://myqmp.anglia.ac.uk/perception5/open.php?customerid=Perception Certificate error Course modules: Final Project ... Sem1_class2_Ethics.pdf: Final P... Questionmark Perception

Oct 26 2017 | Logged in as : Joshua Walker 1503648

Research Ethics Quiz for Health Sciences etc

Assessment Feedback

Congratulations, Joshua Walker 1503648

You scored 7 out of 10 (70 per cent) at 09:38 on Thursday, 26 October, 2017

You have **PASSED** the Research Ethics Quiz.

Please take a screen capture of this information to attach to your Ethics Approval Application for submission to the relevant Research Ethics Panel.

You can also review how well you did on individual questions by scrolling down to view any feedback.

Total score: 7 out of 10, 70%
Question Feedback
1 of 10
1. Which of the following is most likely to raise a significant ethical issue? (select one).
a) A senior manager in a mental health charity is doing a postgraduate degree and asks his junior staff to complete a questionnaire on workplace wellbeing.
b) An Undergraduate hands out questionnaires about student experience and well-being as students enter the university with a box at reception for their anonymous return.
c) A Biology student is researching which flowers Victorian and Edwardian botanists chose to bring back to the UK.

1 out of 1

You chose the correct answer:

a) A Senior Manager in a mental health charity is doing a postgraduate degree and askes his junior staff to complete a questionnaire on workplace well being.

There is a power relationship between the researcher (the Manager) and the participants (junior staff), which may mean that the consent is not fully voluntary.