Department of Computer Science

Submitted in part fulfilment for the degree of BSc.

Machine Learning to Improve Argumentation

Joel Fright

7th May 2020

Supervisor: Tommy Yuan

Acknowledgements

I would like to thank Tommy Yuan for his help throughout this project, as well as my family and friends for their continued support.

Contents

List of Figures

Executive Summary

0.1 Aims

The aim of this project was to explore the means of implementing machine learning within argumentation theory. This means being able to generate an argument between two agents by which the agents progressively improve in argumentation skill as more simulations of the dialogue are run and the Q table is built and expanded upon.

0.2 Motivation

My motivation is primarily to gain a better general understanding of how machine learning works and then to apply that to a more specific field, that being, argumentation theory. I would like to adapt and increase abstraction on the current systems available. This might be by reducing the level of argumentation theory such that it consists of fewer constituent parts as well as providing a more general answer to machine learning with the topic. By doing this I will have shown that there is a possibility to understand and apply arguments to simpler forms of machine learning, proving that argumentation can be applied to many fields of computer science whilst not over complicating the problem.

0.3 Methods

The overall basis of the methodology was to build up the argument such that it was coherent and stayed relevant to the argument domain which was, in this case, capital punishment. Then I was to move on to applying this argument into a machine learning algorithm, Q-learning. This meant that initially I had to build up the foundations on which my algorithm would lie, ultimately deciding on a layout consisting of proposals, counter-proposals, questioning and evidence. This was more abstract than some structures I had looked at and less abstract than others, providing a nice balance.

I then began work on implementing Q tables for each agent as well as the Q-learning algorithm. The algorithm in its most basic form was already set in stone from various papers, it was simply a case of applying it to my specific problem and domain. The same could be said for the Q tables, which proved rather difficult to apply to the field of argumentation, as a dialogue is hard to represent literally. Nevertheless, upon completion of the machine learning algorithm, I now had two agents arguing against each other whilst simultaneously learning the best methods by which to argue within the given domain.

0.4 Results

I was able to see from the final algorithm that the Q tables did indeed improve over time and included asymptotic behaviour which is very common when considering machine learning problems. This asymptotic behaviour showed that the Q tables had, after a couple of iterations, began to reach a unanimous decision on the best ways to argue within the chosen problem domain.

I also saw that the argument itself, shown by an output window on the interface, did indeed appear coherent and stayed relevant to the domain. This was good because it meant that the argumentation theory abstraction that I had provided the two agents with was enough for them to have a well structured argument based dialogue.

It was concluded that the program ran as it should have, however more could be done to improve the level of argumentation, such as providing more advanced propositional logic, applying it to a neural network and much more.

0.5 Issues

Due to the nature of the project, there are no legal, social, professional, or commercial issues. However, ethical issues may arise with some of the information put forward to the knowledge base, but seeing as this is not for public use I do not believe this is a problem.

1 Introduction

1.1 Overview

Argumentation is becoming increasingly more fascinating and alluring topic within computer science. Alongside the already very established machine learning, we can begin to develop ideas that allow us to expand our knowledge of argumentation beyond what we currently know today. The goal of this project is to apply machine learning (specifically reinforcement learning) to argumentation to improve an agents ability to argue and persuade by allowing them to interact with the another agent and gain valuable information on how to argue effectively.

1.2 Argumentation

As defined by Eemeran in [1], “argumentation is a verbal, social, and rational activity aimed at convincing a reasonable critic of the acceptability of a standpoint by putting forward a constellation of propositions justifying or refuting the proposition expressed in the standpoint.” When analysing argumentation we have to take in a plethora of factors, as the concept itself is vast and complex. The initial problem that we encounter happens to be how we define what a good argument entails, this is a key problem. There are many different approaches to this, Carenini [2], suggests quantising the values and preferences, then evaluating those discrete values by classing them and applying specific functions.

The current system as defined by Yuan [3], uses a selection of types of dialogue strategies that one may encounter within an argument. These include assertions; questions; challenges; withdrawals and resolution demands. Along with this, we use propositional logic in order to determine the strength of an argument, and whether it makes sense in the context given.

1.3 Machine Learning

Machine Learning is defined very briefly by Mitchell [4] as the means by which a program will learn from experience with respect to some class' of tasks and a performance measure, if its performance in said tasks, improves with experience. In this specific case we will be delving into the world of reinforcement learning, where we will be able to apply the Q-learning algorithm to argumentation such that we can improve the agent's strategies and techniques in regards to how they argue, consequently improving the computers persuasion abilities and allowing for the agents to gain a better understanding of the knowledge base.

1.4 Project Aims and Objectives

The overall aim for this project personally is to gain a better understanding of how argumentation can be applied to machine learning in order to improve said argumentation. More specifically however, the aim of this project is to pit two agents against each other in an argument based scenario and allow them to learn through the use of Q-learning. I will then evaluate the effectiveness of these arguments with different evaluative algorithms. The final goal is to have a machine learning AI that will argue with another agent in a logical and coherent way, whilst both agents improve their understanding of how argumentation works and how arguments are formed, structured and ultimately won.

1.5 Project Overview

Within this project I will first discuss the literature around this topic, including current and past projects on argumentation, as well as discussing about machine learning and how it can be applied to argumentation. I will then move onto a problem analysis where I discuss how I would define a good system and what I should include to ensure that the program is functional and efficient, as well as including the necessary tasks that need to be completed in order for this project to be considered a success. This will be in the form of a list and will have justifications for every requirement of the project underneath. Then I will move on to the methodology, which will be written in sequence with the code to show the linear progression. Here I will define how I did everything and why I chose to do it in that manner. I will then discuss the results derived from my project and evaluate how they may be improved and draw my conclusions of the project, summarising how I felt it went and what I have learnt.

2 Literature Review

2.1 Argumentation

Argumentation has been around for many many years, therefore it has amassed a lot of attention and, consequently, a lot of research. Some earlier models of argumentation include Toulmin's Model [5] whereby the basis of the argument is that a person makes a claim, gives grounds to support said claim and finally backs the grounds with a warrant. Since then argumentation theory has expanded and developed, with an increased look into the logic of the system. As discussed prior, Yuan, Moore and Grierson [3], have defined a model of argumentation which contains a set of rules and propositional logic that will eventually help the computer to understand how an argument is structured.

We can see from the wealth of information available that machine learning is a large factor within argumentation. Toni and Cocarascu [6], discuss the future of argumentation within machine learning, as well as talking about the various different approaches to the topic, such as *ABML*, *MAICL* etc. Within this discussion we discover that all the approaches have their own advantages and disadvantages, suggesting that the topic is still very new and there is no clear cut answer to argumentation.

Furthermore, argumentation can be modelled in various different ways, some examples such as *Argumento+* in [7] uses a very abstract view of argumentation, whittling it down to a graph like structure that has a very simple reward function, whilst not including any dialogue that one might see in a real world scenario. However, there still existed a very evident generalisation for the dialogue within this abstracted argumentation scenario.

More advanced and complex examples include Yuan's human-computer dialogue system [3], whereby a user is made to argue against a computer agent by the use of propositional logic as well as a strict and more computationally complex dialogue system, including a variety of different move types and rules by which both the user and the agent must follow. This system of argumentation and game design truly shows how far the field of argumentation can stretch, whilst at the same time, only scratching the surface of the potential of argumentation theory.

2.2 Reinforcement Learning with a Focus on Q-Learning

Copious amounts of research has been done in the field of Q-learning, as it is a vital element of machine learning and specifically reinforcement learning. Alahmari [8], has created an abstraction of an argumentation system with the use of Q-learning, whereby a game is created on the basis of a few rules and including human interaction required to play against the agent. Furthermore, Q-learning is an element within the broader topic of machine learning. Toni and Cocarascu [6], dive into the other two main areas of machine learning, unsupervised and supervised learning. They begin to define abstractions of how argumentation may be represented in said fields, however delving into these two topics prompts a wealth of discussion which would be redundant in progressing our understanding for this project.

```
Initialize $Q(s, a)$ arbitrarily
Repeat (for each episode):
   Initialize $s$
   Repeat (for each step of episode):
      Choose $a$ from $s$ using policy derived from $Q$ (e.g., $\varepsilon$-greedy)
      Take action $a$, observe $r$, $s'$
      $Q(s,a) \leftarrow Q(s,a) + \alpha\big[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\big]$
      $s \leftarrow s'$;
   until $s$ is terminal
```

Figure 2.1: Q Learning Algorithm

The figure 2.1 shows how Q-learning uses a state and action pair to explore and route the optimal path through a series of episodes. This idea was first introduced by Watkins in 1989 in his Ph.D thesis [9]. The convergence was then later proven by Watkins and Dayan in their 1992 paper titled Q-learning [10], in which they describe Q-learning as the ability to provide "agents with the capability of learning to act optimally in Markovian domains by experiencing the consequences of actions".

Q-learning is most predominately used in the field of machine learning due to it's practicality. The algorithm itself is a model-free based algorithm meaning that the reward and delta values are not needed to be known prior in order to generalise and explore the environment. Instead the learning consists of observing the reward and updating the next q value accordingly.

2.3 Reinforcement Learning for Argumentation

Within machine learning it is important to evaluate the effectiveness of the Q-learning algorithm. It is important to evaluate the effectiveness as it ensures that the Q Learning algorithm is doing what it is supposed to. Upon gathering an understanding of how well the system is working, we are then able to form conclusions on the project as a whole. There are many way to do this, one of the simpler heuristics is to look at the number of moves it took to reach resolve. Some other examples of possible heuristics to diagnose effectiveness within argumentation include analysing which side won the argument, how many moves it took for them to win, whether the dialogue followed a coherent pattern as we know today and much more. Looking at the winning heuristic, many previous examples of this problem use the reward average over the games dialogue to decide on the winner of the argument. This can be done by analysing the Q table and calculating and average of all the values per agent.

Furthermore, another heuristic mentioned was evaluating the number of moves that was required to win. Alahmari in [11] stated that one of the aims is "to motivate the agent to win the dialogue game with the minimum number of moves. I believe this is key in the machine learning automation, as it can be said that the less moves it takes for an agent to convince the opponent or 'win the game', the more effective the agent was at arguing.

Some other more complex heuristics include analysing whether the dialogue had relevancy, meaning deciding whether the two agents were talking in a manor that stayed in line with the argument domain, as well as coherency, which evaluates if the agents dialogue made sense in terms of the logical structure of an argument. Within Alahmari's thesis [11] there is a whole chapter dedicated to coherence and relevance, aptly named the 'quality of the dialogue', within this there is a discussion about how to measure this topic that is inherently continuous, some deeper analysis suggests using a formula based around incoherency whereby the number contradictions an agent makes is divided by the number of moves the same agent has made. This is further expanded on with an analysis of agent irrelevance being the number of times focus is switched divided by the number of moves taken.

In addition to this, Alahmari [11] defines two levels of abstraction, a game called *Argumento+*, and a higher level propositional logic based dialogue game. Within the latter we are shown the design for the knowledge base architecture, which can be seen in figure 2.2, in which we can see how the data from the knowledge base feeds into the agents understanding of how to structure an argument.

As well as the structure of the argument, it is important to consider the content as well. The domain that I have decided on is the issue of capital punishment, as I feel this is a very polarising topic, meaning that both agents will have a fair amount to both propose and counter-propose. It was also the example used in Yuan's paper [3], therefore I can assume that there is a high level of validity in using it as an argument domain. It is in fact from this example that I shall build the knowledge base upon, as I believe it is valid and will allow me to focus on more complex aspects of the project. This means that all of the syntax can be and has been interpreted prior so I can assume that not only can it definitely be presented to a machine learning environment, but also that the argument is correct syntactically. The domain discussed uses the validity of the source that the information has derived from as the initial reward values. This works well within the system as it means that when an agent uses a more reputable source it is commended higher, which is how argumentation theory tends to work in real arguments.

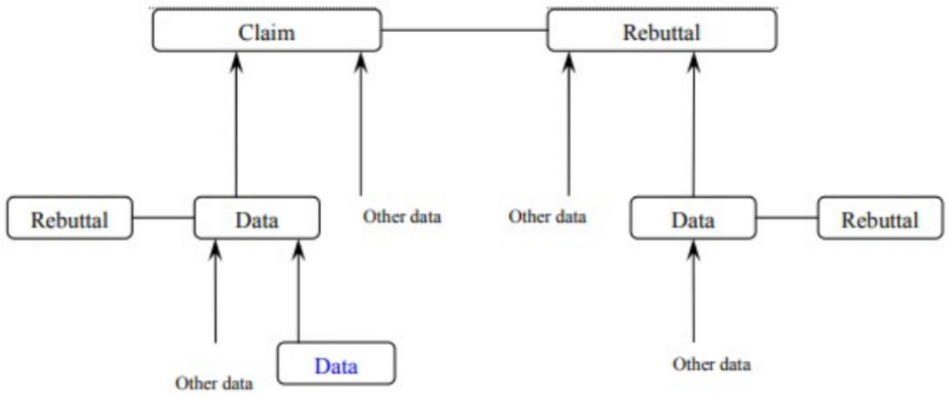


Figure 2.2: Knowledge Base Architecture

2.4 Summary

As shown, there are many ways that I could approach this project, some decidedly better than others for my specific needs. My main goal is to maintain a certain level of abstraction therefore avoiding complicating the very basis that I am trying to achieve. Much of the discussed literature is very advanced but nevertheless will help me with my understanding and consequently progress me towards my end goal.

3 Problem Analysis

I have decided to lay out some well defined aims and goals so that upon nearing the end of my project I can reflect and evaluate what I have succeeded and failed in and decide whether or not I believe the project was a success. The aims however will be quite abstract, the methodology will delve into more detail as to how these specific problems are approached.

1. To have implemented Q Learning

   As the title is 'Machine Learning to Improve Argumentation Skills' I feel it is of paramount importance to have explored the machine learning aspect, therefore, this is my first and primary aim. I believe that in order to have successfully achieved this aim I must have used some variant of the algorithm seen in figure 2.1. We have seen that this is important in the literature review, and have diagnosed why it is that Q-learning is the best algorithm for this specific problem. This is because the system is structured such that there is a state and action pair within every move. Meaning that alongside the algorithm, we are able to efficiently and correctly calculate certain reward values and update those based on the specific moves of the agents. This in turn will improve performance of the agents and is why it is key to include within the project.

2. To have implemented argumentation

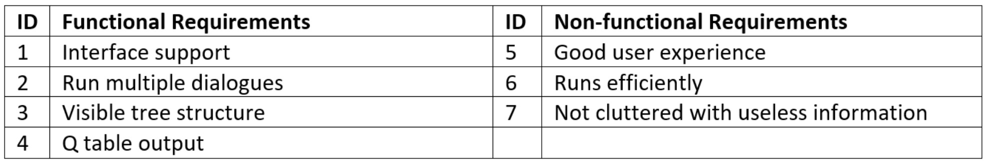
   Coinciding with the previous aim, argumentation is a vital element in piecing the puzzle together. In order to have fully achieved this I believe I must have used a least an certain level of abstraction of argumentation that has been discussed in the current system [3]. This means I need the two agents to argue logically such that the grounds of argumentation theory are maintained. In detail, the game I am looking to integrate within my work is one of a higher level of abstraction than *Argumento+* [7], but lower than Yuan's [3]. I believe this area of argumentation is well suited for my needs and will consist of a basic argumentation knowledge base, taking some inspiration from Toulmin's model [5] in terms of the move types involved. Therefore the game must have two agents selecting certain possible move types that are predefined whilst applying the correct information associated with that move type, taken from the knowledge base. This dialogue must then run until the agents have completed their chosen proposals. In addition, whilst this is happening the Q-learning algorithm must be running on the system, building up the q table.

3. To have made a system that is fully functional

   This may seem rudimental, however a fully functional system is key in my belief of a successful project. The end product must have the features as proposed prior, whilst still being able to be operated by a user without fault or confusion. This must be done by use of a clear and coherent user interface, with a back end that completes tasks correctly and efficiently. Below in figure 3.1 we can see the functional and non-functional requirements of the system that I have chosen, some being more obvious than others.

4. To be able to evaluate the effectiveness of the Q Learning

   In order to know that the methodology you have used is working it is vital to evaluate it. Therefore for this aim I must have evaluated whether my Q-learning algorithm has done what I want it to do. There are many different heuristics to choose from, analysis which is the best will also be key in ensuring I have completed this fully. The main heuristics I plan to use will be analysis of the average reward which will be seen within the Q table itself, as well as analysing coherency and relevancy by observing the interactions between the two agents.



ID	Functional Requirements	ID	Non-functional Requirements
1	Interface support	5	Good user experience
2	Run multiple dialogues	6	Runs efficiently
3	Visible tree structure	7	Not cluttered with useless information
4	Q table output		

Figure 3.1: Functional and Non-functional Requirements

The starting point for this system is to create a system that has the basis of argumentation already ingrained, this could be done by applying a knowledge base to the system such that the two agents have some inputs, which they can handle however need be. The process will then be to develop this to enable machine learning, and in particular, Q-learning such that the two agents begin to learn from each other as well as from the dialogue itself and can begin to argue more cleverly. Finally, an evaluation must be done on how the program has performed, via a set of heuristics that assess efficiency and generalisation.

4 Methodology

4.1 Tools Used

For this project I used Java with the IntelliJ IDEA IDE. This is because Java was recommended by my supervisor as the language to use for this project, as well as it being the language I am most comfortable in. Matlab was used to plot and analyse the data. I also used Git and GitHub as a method of version control.

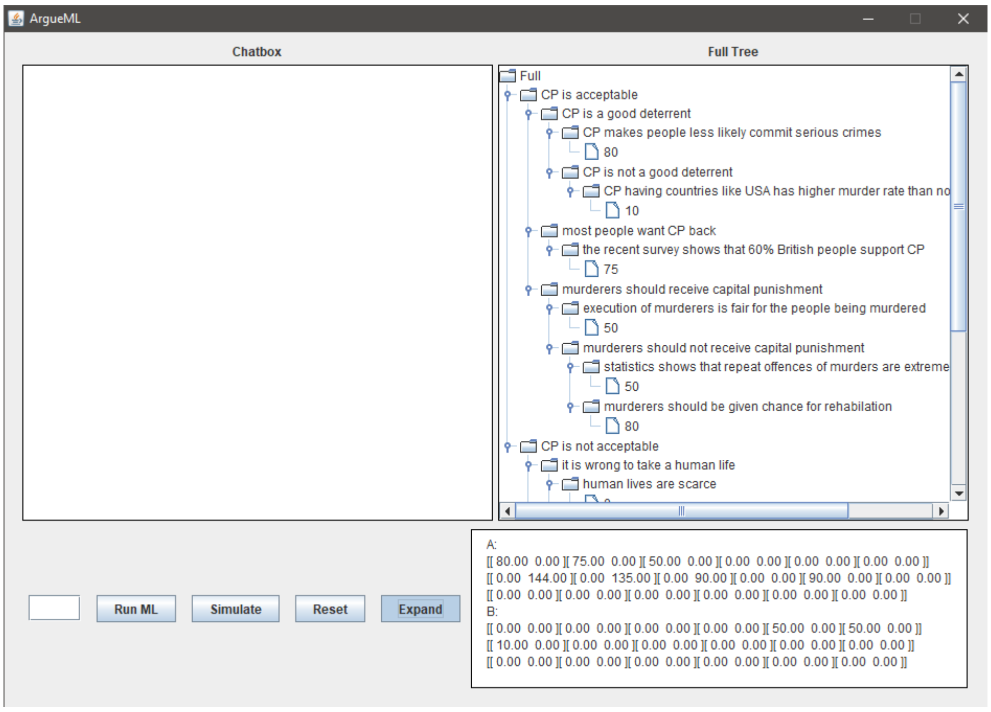
4.2 The UI

My plan for the UI was relatively simple, therefore using JSwing components seemed most appropriate. The left hand side includes an output box (JTextField) where I am able to see clearly what the program is outputting, such to decide whether it is doing what I have asked it to do within the code.

Furthermore, on the right hand side there is the tree based structure that I use to represent my knowledge base and argument design. This is used primarily for debugging so that I can reference back to it to test to see if the agents are following the correct paths, and that every path is being explored and utilised. The tree structure that has been implemented is a static version, of which the agents can create their own abstracted trees which include the debate history from the actions they have taken. As well as these features we are able to see the Q Table displayed on the interface, this allows me to see if the dialogue is performing correctly and all values are being inputted into the Q Table in the right order.

Finally, there was need for buttons to begin and end the automation, this includes a 'step' like button which runs a single argument, as well as a multi-run feature where I can input a selected number of iterations and then ask the computer to run them all at once. This is important because it means I can train my algorithm on a variety of different cases and therefore evaluate the effectiveness of said algorithm. There are also some added features to allow for ease of use within the UI which links to problem 3 in the analysis stage. Below in figure 4.1 we can see an example of what this

UI looks like.



```
Full
  CP is acceptable
    CP is a good deterrent
      CP makes people less likely commit serious crimes
        80
      CP is not a good deterrent
        CP having countries like USA has higher murder rate than no
          10
    most people want CP back
      the recent survey shows that 60% British people support CP
        75
    murderers should receive capital punishment
      execution of murderers is fair for the people being murdered
        50
      murderers should not receive capital punishment
        statistics shows that repeat offences of murders are extreme
          50
        murderers should be given chance for rehabilitation
          80
  CP is not acceptable
    it is wrong to take a human life
      human lives are scarce
```

```
A:
[[ 80.00  0.00 ][ 75.00  0.00 ][ 50.00  0.00 ][ 0.00  0.00 ][ 0.00  0.00 ][ 0.00  0.00 ]]
[[ 0.00  144.00 ][ 0.00  135.00 ][ 0.00  90.00 ][ 0.00  0.00 ][ 90.00  0.00 ][ 0.00  0.00 ]]
[[ 0.00  0.00 ][ 0.00  0.00 ][ 0.00  0.00 ][ 0.00  0.00 ][ 0.00  0.00 ][ 0.00  0.00 ]]
B:
[[ 0.00  0.00 ][ 0.00  0.00 ][ 0.00  0.00 ][ 0.00  0.00 ][ 50.00  0.00 ][ 50.00  0.00 ]]
[[ 10.00  0.00 ][ 0.00  0.00 ][ 0.00  0.00 ][ 0.00  0.00 ][ 0.00  0.00 ][ 0.00  0.00 ]]
[[ 0.00  0.00 ][ 0.00  0.00 ][ 0.00  0.00 ][ 0.00  0.00 ][ 0.00  0.00 ][ 0.00  0.00 ]]
```

Figure 4.1: UI Design

4.3 Receiving the Argument Domain

In order to make the argument structure I first had to define a knowledge base. By using an example of the current system I was able to extract useful information in order to build this up under the domain of capital punishment. By feeding this into a comma separated file I was able to allow the agents to interpret the information given, through the use of a buffer reader. The comma separated file was structured such that it had the argument case, which defined what the agent would be saying, for instance, *"a state has the right to deliberately execute the wrong people"*. This part was important in the evaluative process so that we were able to see that the outputted argument made logical sense.

Alongside this, there was a variable declaring whether or not it was for or against capital punishment. There are three more rows currently included however I will come to those later. Nevertheless, using these first two rows I could already build up a very basic argument. In order to complete requirement 2 within the problem analysis it needed some form of structure, as argumentation is not simply a variety of disorganised strings of statements but instead a much more complex linguistic style.

4.4 Structuring the Argument

Referring back to figure 2.2 we can see that the integral structure is already outlined. The key part of this process was to design a way to represent this and other data structures within the code, more specifically, in the java programming language. There are many ways in which this could be done, I comprised a list of possible routes and decided that the best option would be to, by using a *JTree* component as well as *DefaultMutableTreeNodes*, formulate a set of routes by which the agent should follow in order to properly construct an argument.

The first task was to create the tree as seen in figure A.1 and A.2. This was deemed quite a simple task as all it entailed was to add leaf nodes within the code to expand the matrix, whilst using the string variables held in the leaf nodes as output of the agents. I made two trees to represent the two different sides of the argument, for and against. This decision seemed the best thing to do as disparity would be needed in the future. I could then append the rewards to the end of the tree which were later retrieved in the Q-learning process. This structure was optimal for many reasons however there were many difficulties that came along with it. I will discuss these positives and negatives in the evaluation chapter.

As well as this it was important to define the reward factors for each argument, which can be seen as the numbers in figure 4.1, these are based on the reputability of the source that the information from which that dialogue came. As an example, underneath 'the recent survey shows that 60% of British people support CP' we can see that the initial reward value for that statement is 75, which can be attributed to that fact that it was sourced from the BBC, which is a very trustworthy news source and does not have bias due to being paid for by the tax payer. All these rewards have had analysis on their sources to decide their reward value.

4.5 Forming the Argument

The argument was formed in three main constituent parts, the proposal, the question and the evidence. These are the basis of any modern day well constructed argument. I was then able to make a dialogue out of the constituent parts, known as the rebuttal. This was to be used whenever the opposing agent from the claimant had a counter-argument. This is comprised of a counter-proposal and evidence to back up said counter-proposal, allowing for a more dynamic range of argumentation with an expanded number of routes and possibilities available for each agent. We can then see that this structure has been designed to fit the knowledge base architecture in figure 2.2. This way of laying out the argument has major similarities to another proposed argumentation strategy know as Toulmin's model of argumentation [5], with a slight layer of abstraction in order to increase simplicity, as due to the nature of the project, this is an abstracted and more simplified example of what can be achieved.

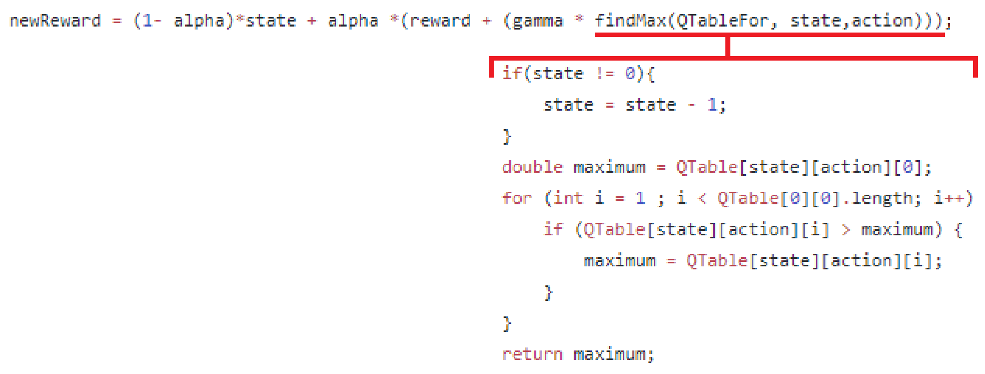
The next objective was then to have each agent call certain nodes from the tree in an order that made logical sense. This could be done fairly simply as the agents would follow the path on the tree until reaching an end node and then restarting with a different initial proposition, however there were some difficulties. I needed to ensure that the agents did not repeat proposals, such that the q tables was not built up on false dialogues. In order to do this I had to take each primary proposal and set a boolean value to true or false suggesting as to whether or not it had been expanded.

4.6 Implementing the Q Table

Implementing the q tables is arguably one of the harder if not hardest aspect of the entire project. This fundamentally derives from the fact that representing a dialogue system in which two agents are participating is very complex. The major difficulties were deciding whether or not to have two separated q tables and how to represent rebuttals as they appear in the oppositions side of the tree. It was settled that the q table would be two tables each containing all the possible nodes for both users, with each node representing a state ie. a proposal, counter proposal or evidence, and the nodes within that representing the actions. This then formed the base structure of a 3 dimensional table, with the main table being 6 columns by 3 rows and each one of those elements in that containing 2 nodes of possible actions available to the agent.

The states were defined as being the current move from the agent that had played previously, which was selected from a node within the tree seen in figures A.1 and A.2. The actions were then defined as the two possible move types, proposal or evidence. The reward function was then built up based on this state and action pair, and can be seen in figure 2.1, alongside being included in the code as seen in figure 4.2.

Then for initial testing I set randomness values at every turn such that the dialogue could explore all nodes. I then compared the results of the dialogue with the q tables to make certain that the q table's structures was being properly interpreted by the code. Subsequently, I had to start initialising the q tables the correct way with the chosen Q-learning algorithm.



```
newReward = (1- alpha)*state + alpha *(reward + (gamma * findMax(QTableFor, state,action)));
```

```
if(state != 0){
    state = state - 1;
}
double maximum = QTable[state][action][0];
for (int i = 1 ; i < QTable[0][0].length; i++)
    if (QTable[state][action][i] > maximum) {
        maximum = QTable[state][action][i];
    }
}
return maximum;
```

Figure 4.2: Reward Function

4.7 Q-Learning

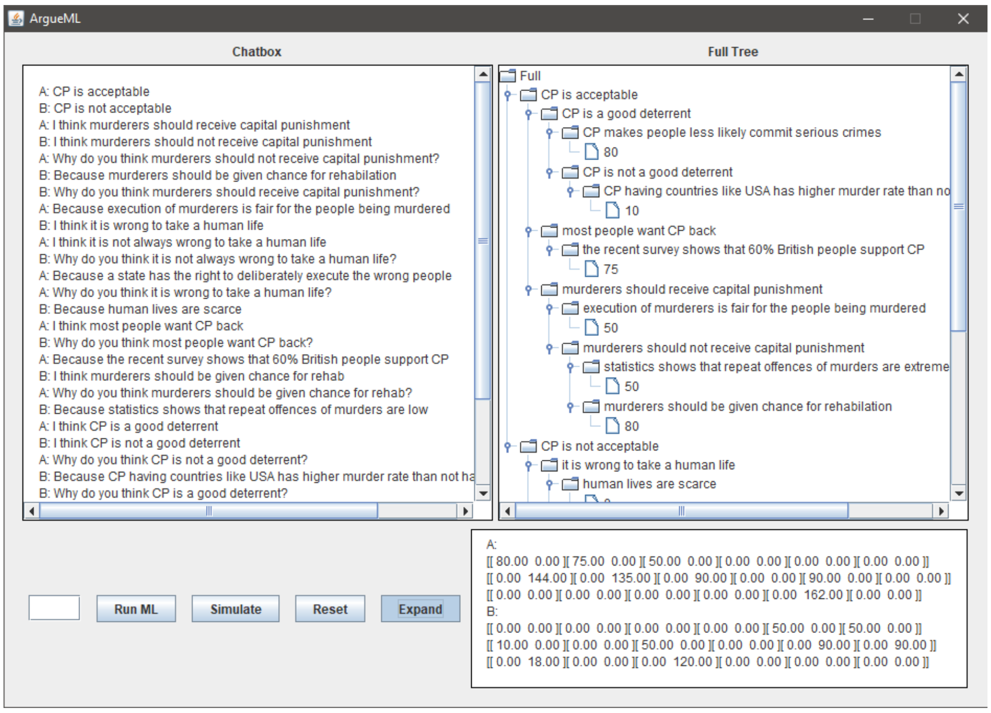
I have already defined the Q-learning algorithm in the previous chapter, as can be seen in figure 2.1. It was now simply a matter of implementing the algorithm within the scope of the project. This meant manipulating certain parts of my current project to fit the algorithm and vice versa. I started by defining the algorithm as it is stated in the figure as well as how it is explained in Watkins and Dayan's paper [10]. Once I had implemented this basic structure I could then expand on it, which included linking it to the q tables for each agent. The way the q tables were represented meant that finding maximum values was a fairly simply algorithm, which in turn made the Q-learning algorithm rather efficient.

4.8 Evaluation of Learning

In order to evaluate the learning I first had to decide on my control variables, this mainly meant I had to decide on how many arguments to run the learning on. I decided on 25 arguments because when testing I found that the q tables appear to regress at around this number for a variety of different learning rates. I then had to decide on the increments that I was going to change the learning rates, a value of 0.2 seemed most appropriate as it covered a lot of bases for 0.5 to 0.9. I then wrote the average reward to a csv file alongside the number of iterations at that average reward. I took this output and was able to represent it in Matlab by using a line graph.

5 Results and Evaluation

With my project being a simplified and abstracted version of something that is, in nature, quite complex, I had to reduce and remove a lot of factors that allowed for more advanced techniques. This therefore was the biggest problem within the project. However, despite that I do believe I have attained a good and well thought-out demonstration of what I was trying to achieve at the beginning of the project.



```
A: CP is acceptable
B: CP is not acceptable
A: I think murderers should receive capital punishment
B: I think murderers should not receive capital punishment
A: Why do you think murderers should not receive capital punishment?
B: Because murderers should be given chance for rehabilation
B: Why do you think murderers should receive capital punishment?
A: Because execution of murderers is fair for the people being murdered
B: I think it is wrong to take a human life
A: I think it is not always wrong to take a human life
B: Why do you think it is not always wrong to take a human life?
A: Because a state has the right to deliberately execute the wrong people
A: Why do you think it is wrong to take a human life?
B: Because human lives are scarce
A: I think most people want CP back
B: Why do you think most people want CP back?
A: Because the recent survey shows that 60% British people support CP
B: I think murderers should be given chance for rehab
A: Why do you think murderers should be given chance for rehab?
B: Because statistics shows that repeat offences of murders are low
A: I think CP is a good deterrent
B: I think CP is not a good deterrent
A: Why do you think CP is not a good deterrent?
B: Because CP having countries like USA has higher murder rate than not ha
B: Why do you think CP is a good deterrent?
```

```
Full
  CP is acceptable
    CP is a good deterrent
      CP makes people less likely commit serious crimes
        80
      CP is not a good deterrent
        CP having countries like USA has higher murder rate than no
          10
    most people want CP back
      the recent survey shows that 60% British people support CP
        75
    murderers should receive capital punishment
      execution of murderers is fair for the people being murdered
        50
      murderers should not receive capital punishment
        statistics shows that repeat offences of murders are extreme
          50
        murderers should be given chance for rehabilation
          80
  CP is not acceptable
    it is wrong to take a human life
      human lives are scarce
```

```
A:
[[ 80.00  0.00 ][ 75.00  0.00 ][ 50.00  0.00 ][ 0.00  0.00 ][ 0.00  0.00 ][ 0.00  0.00 ]]
[[ 0.00  144.00 ][ 0.00  135.00 ][ 0.00  90.00 ][ 0.00  0.00 ][ 90.00  0.00 ][ 0.00  0.00 ]]
[[ 0.00  0.00 ][ 0.00  0.00 ][ 0.00  0.00 ][ 0.00  0.00 ][ 0.00  162.00 ][ 0.00  0.00 ]]
B:
[[ 0.00  0.00 ][ 0.00  0.00 ][ 0.00  0.00 ][ 0.00  0.00 ][ 50.00  0.00 ][ 50.00  0.00 ]]
[[ 10.00  0.00 ][ 0.00  0.00 ][ 50.00  0.00 ][ 0.00  0.00 ][ 0.00  90.00 ][ 0.00  90.00 ]]
[[ 0.00  18.00 ][ 0.00  0.00 ][ 0.00  120.00 ][ 0.00  0.00 ][ 0.00  0.00 ][ 0.00  0.00 ]]
```

Figure 5.1: The Full Argument

Upon setting the algorithm to a state whereby the two agents deliberately argue the entire knowledge base on the first dialogue by maxing out the exploration rate, we would be able to see an example of the dialogue in its full form, as shown in figure 5.1. From looking at this dialogue we can pick out some key elements. For example, we can see that both agents argue logically and coherently. We can see there are elements of proposals, questioning, giving of evidence, as well as counter-proposals and all their constituent parts. This is obviously not how a traditional dialogue would look like as not all the nodes would be expanded like this, but it does show us that the dialogue is functional, and argumentation theory is being maintained.

Some, if not most, difficulties arose from the machine learning element of the project. The biggest problem I encountered was how to combine certain elements of argumentation with certain elements of Q-learning in a way that was the most sensible. Ultimately I do believe my representation was the best that I could produce within the time frame, as it met itself in the middle on a scale of complexity and efficiency.

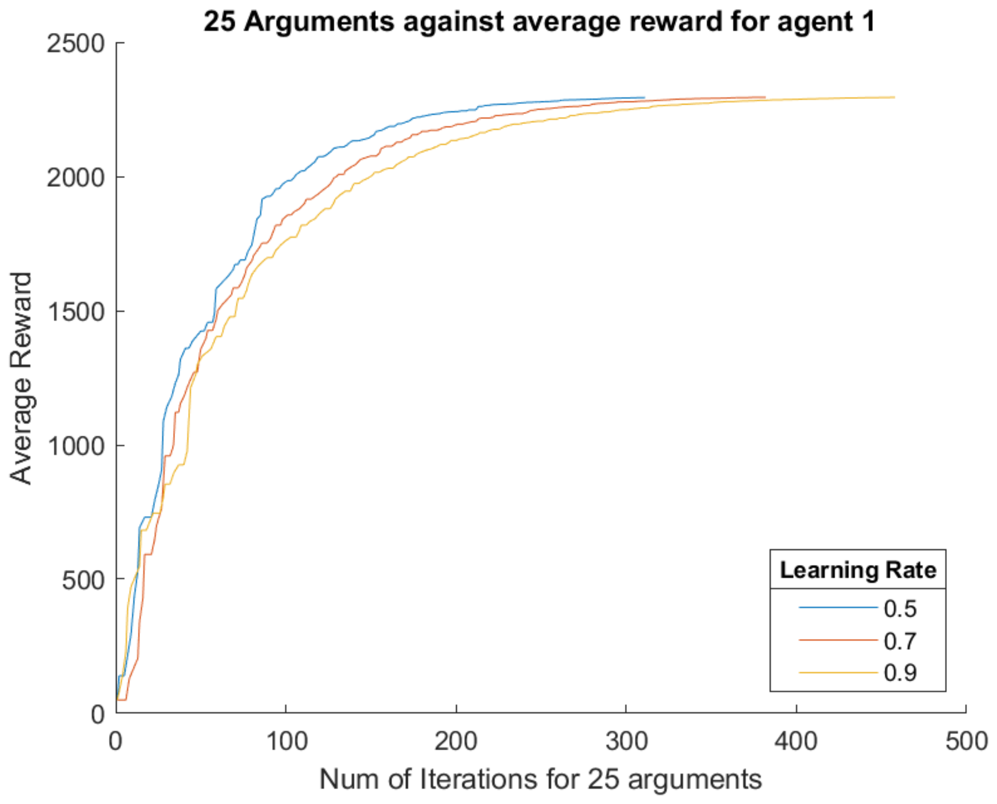


Figure 5.2: Iterations vs Average Reward (Agent 1)

In order to evaluate the system, I decided to run three different iterations of 25 arguments against average reward. The x-axis is therefore each action for the dialogues run, or in other words, each proposal, evidence or counter-proposal over the course of 25 sample arguments. The y-axis refers to the average reward which is given by averaging the values within the q tables after an action had taken place. The different colours of the figures 5.2 and 5.3 represent the learning and exploring rates, which shows how much the agent is allowed to explore the knowledge base for every move.

From these figures we are able to deduce that firstly, with a higher learning rate, the number of iterations is greatly increased, as can be seen from how far each line extends in the figures. This is because with a higher the learning rate the longer the dialogue takes to explore all the possible options. We can also see that a part from some anomalies the lower the learning rate the faster two agents learn, with them peaking earlier in most cases. Finally, we can see for definite that there is generalisation, as all of the line plots exponentially decrease and plateau over time, this is to suggest that the q tables are correct and fully filled out meaning that the agent now knows the optimal paths and can take the necessary steps to argue effectively and efficiently against the opposition.

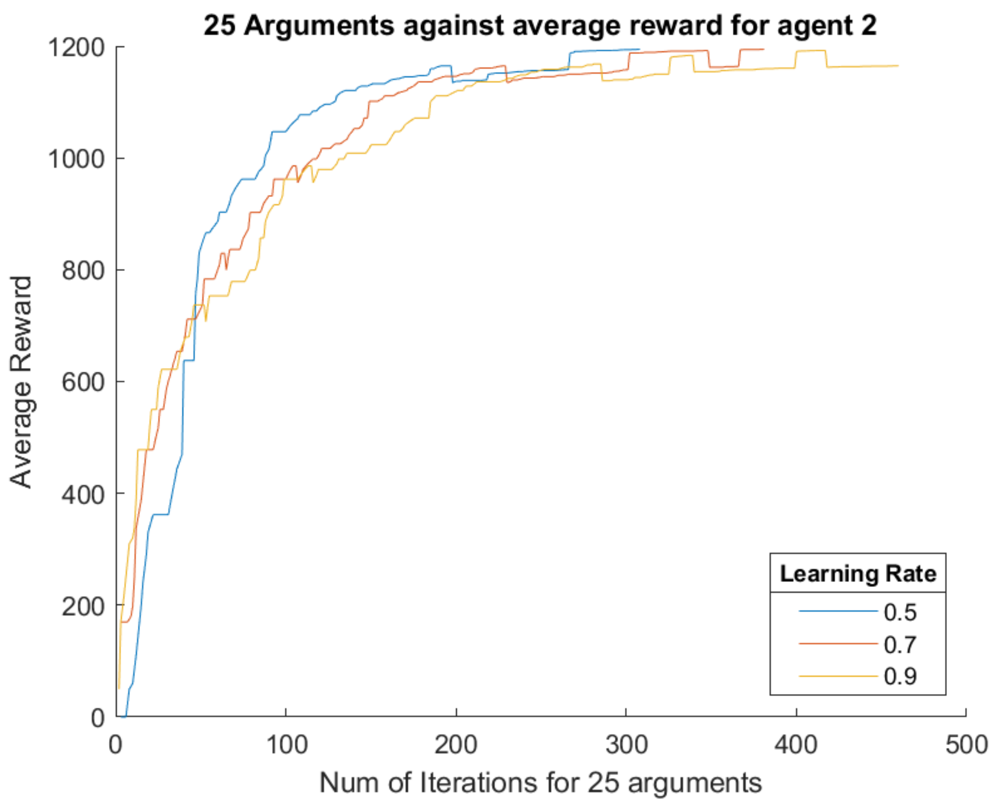


Figure 5.3: Iterations vs Average Reward (Agent 2)

We can see from figure 5.3 that the lines seem to be a bit more messy than the lines in 5.2, this could be because within the knowledge base architecture, agent 1 has a higher total maximum reward which is not ideal, this theory is backed up by the fact that the architecture peaks at 1200, which is much lower than agent 1's 2250. If I were to make any modifications to this project I would firstly make sure that both agents have the same maximum to make the argument more fair overall. This could be done by gaining information for the knowledge base from both sides using the same source.

6 Conclusion

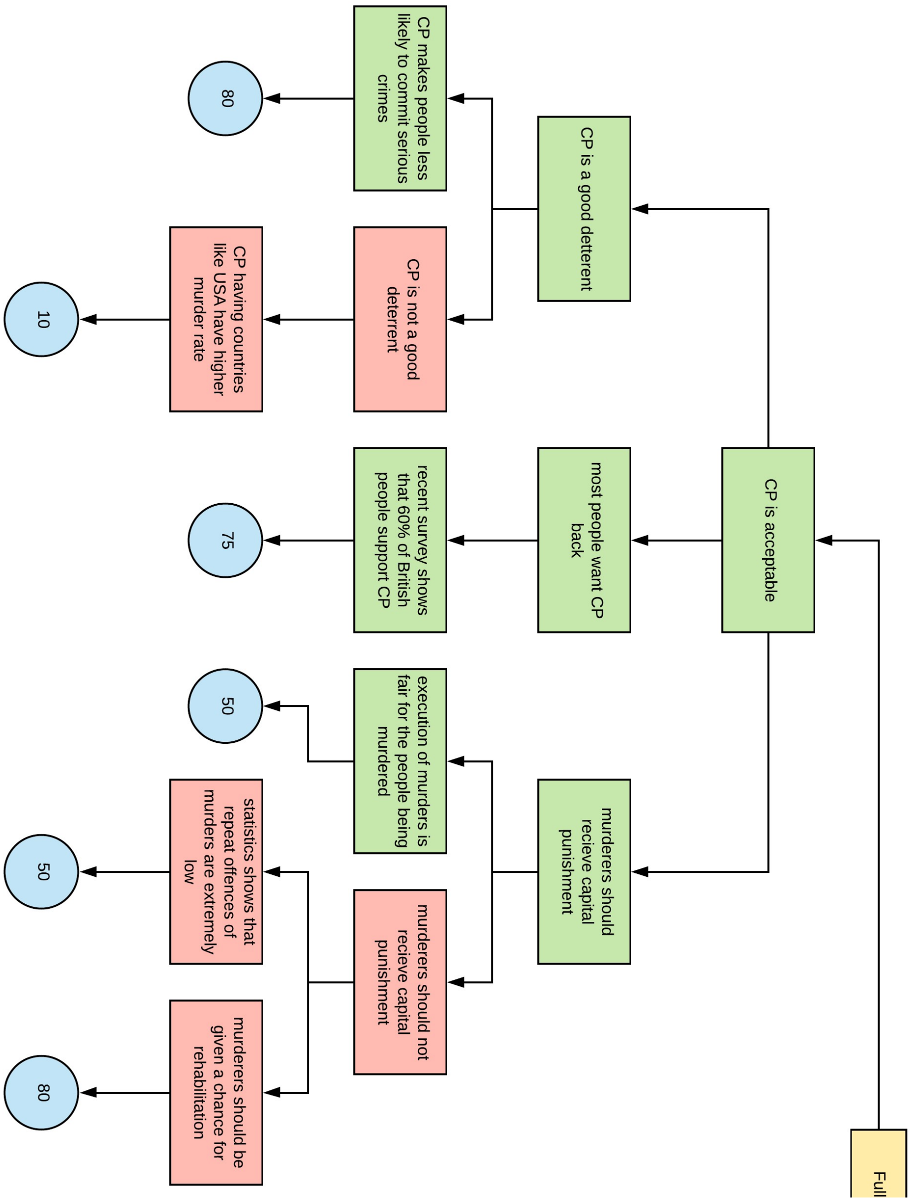
To conclude, we can deduce from the project that argumentation can indeed be used within machine learning and that, by creating an abstract version, there are many different layers and paths of difficulty that can be traversed to increase the level of argumentation theory that can be represented. The results provided conclusive information that when pitting two agents against each other and applying a Q-learning function you can build a system in which the dialogue learns from it's counter-part, which then results in more effective dialogue strategies being used by the agents.

We also saw that the two agents behaved asymptotically, which is quite standard within machine learning, allowing us to gain a better judgement of whether the agents were doing the correct process within the code. However an improvement could have been made in ensuring that the curvature of the second agent was more clear like it was with the first agent.

Some improvements could lie in addressing the appropriate ways to analyse and represent the state and action pair. With the tools I gave myself I feel like I achieved the best I could at the time, however I realise and understand that altering the q tables to be more similar to the actual problem domain would improve the success of the project. There might also be hope in exploring neural networks within argumentation and attempting to improve the level of automation involved in the process.

Overall, I believe the project was a success, I was able to address all the problems that were listed in the problem analysis to a certain degree. However I do believe more could be done to raise the level of complexity within the system, but the time limit refrained me from doing so. If I was to come back to this project later on I would begin to implement more knowledge bases and test on a variety of different cases, as well as test some other machine learning algorithms and see how they performed.

A Appendix



Full
CP is acceptable
CP is a good detterent
CP makes people less likely to commit serious crimes
80
CP is not a good deterrent
CP having countries like USA have higher murder rate
10
most people want CP back
recent survey shows that 60% of British people support CP
75
murderers should recieve capital punishment
execution of murders is fair for the people being murdered
50
murderers should not recieve capital punishment
statistics shows that repeat offences of murders are extremely low
50
murderers should be given a chance for rehabilitation
80

Figure A.1: Tree for Knowledge Base (For CP)

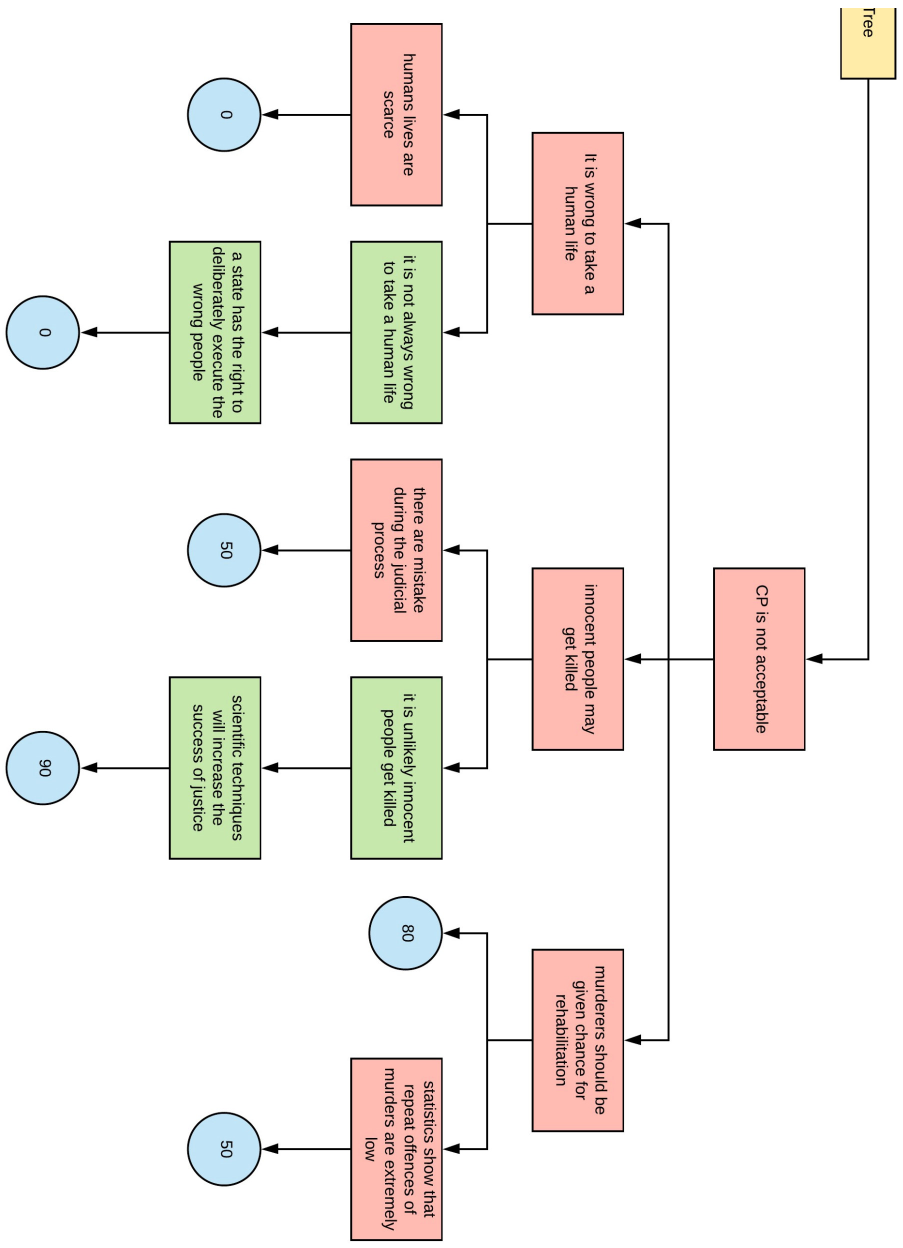

Tree
CP is not acceptable
It is wrong to take a human life
innocent people may get killed
murderers should be given chance for rehabilitation
humans lives are scarce
it is not always wrong to take a human life
there are mistake during the judicial process
it is unlikely innocent people get killed
statistics show that repeat offences of murders are extremely low
0
a state has the right to deliberately execute the wrong people
0
50
scientific techniques will increase the success of justice
90
80
50

Figure A.2: Tree for Knowledge Base (Against CP)

Bibliography

[1] F. Eemeran, R. Grootendorst and F. Henkemans, *A Systematic Theory of Argumentation*. Cambridge University Press, 2004.

[2] G. Carenini and J. Moore, *An Empirical Study of the Influence of User Tailoring on Evaluative Argument Effectiveness*. International Joint Conference of Artificial Intelligence, 2001.

[3] T. Yuan, D. Moore and A. Grierson, *A Human-Computer Dialogue System for Educational Debate: A Computational Dialectics Approach*. International Journal of Artificial Intelligence in Education, 2008.

[4] T. Mitchell, *Machine Learning*. McGraw-Hill, 1997.

[5] J. Karbach, *Using Toulmin's Model of Argumentation*. Journal of Teaching Writing, 1987.

[6] O. Cocarascu and F. Toni, *Argumentation for Machine Learning: A Survey*. IOS Press, 2016.

[7] S. Alahmari, T. Yuan and D. Kudenko, *Policy generalisation in reinforcement learning for abstract argumentation*. Proceedings of the 18th Workshop on Computational Models of Natural Argument, 2018.

[8] ——, *Reinforcement Learning for Abstract Argumentation: A Q-learning approach*. Adaptive and Learning Agents Workshop, 2017.

[9] C. Watkins, *Learning from Delayed Rewards*. Cambridge University, 1989.

[10] C. Watkins and P. Dayan, *Q-Learning*. Springer, 1992.

[11] S. Alahmari, *Reinforcement Learning for Argumentation*. University of York, 2020.