Natural Language Processing and Reinforcement Learning to Generate Morally Aligned Text

What's the optimal weight to maximise morality without affecting performance?

Boudier, K.[1]

Supervisor(s): Pradeep Murukannaiah[1] **, Enrico Liscio & Davide Mambelli**[1]

[1]EEMCS, Delft University of Technology, The Netherlands

Name of the student: Boudier, K.
Final project course: CSE3000 Research Project
Thesis committee: Pradeep Murukannaiah[1] , Enrico Liscio & Davide Mambelli[1] , Jie Yang

Abstract

In our everyday life, people interact more and more with agents. However, these agents often lack a moral sense and prioritize the accomplishment of the given task. In consequence, agents may unknowingly act immorally. There has been a lack of research or progress to endow agents with human morality and an internal sense of right and wrong. As of today, agents have a primitive representation of morality often represented as one value. In contrast, humans have multiple reasons to judge an action as moral. In the hope of creating agents that are imbued with a more complex and human moral, we implement an agent with a pluralist approach to morality. The agent is built within the Jiminy Cricket environment. This preexisting environment has multiple games with diverse scenarios where actions of varying morality need to be made. The objective of our research is to observe if an artificial conscience approach can steer agents toward moral behavior without sacrificing performance. With the help of a genetic algorithm, we determine the importance of morality in comparison to performance.

1 Introduction

It is crucial for Large Language Models (LLMs) like ChatGPT to be imbued with human-like morals. In the past years, there was an increase of technologies relying on Natural Language Processing (NLP). While technologies such as ChatGPT are ever-growing in popularity and used daily, it is critical to have agents that are capable of being moral to prevent any form of harm. Agents may unknowingly act immorally if they are only trained to maximize reward. Furthermore, they could simply ignore moral concerns. This is why ensuring the alignment of LLMs with human morality is a crucial consideration before a widespread implementation and more popular of LLMs.

The implementation of moral agents that try to maximise performance while remaining morally correct is still in the early stages of development. However, it is very hopeful as Hendrycks et al. [6] show that it is possible to create a moral agent that performs as well as non-moral agents for a given task. These agents use the Contextual Action Language Model (CALM) agent architecture [9]. In the research, Hendrycks et al. create an environment called the Jiminy Cricket environment. This environment allows the agent to learn how to act morally through text-based games associated with reinforcement learning. The objective of the agent is to progress as far as possible within the game by taking actions. Each action has two elements associated with them: game progression and morality. A downside of this research is that they consider human morality to be a single value, whereas humans have more than one way of judging an action as moral or immoral. Overall, this research is the precursor to our project as we want to create a more elaborate representation of morality to capture the complex nature of morals.

We intend to investigate whether the Moral Foundation Theory (MFT) approach to moral plurality can be applied to the Jiminy Cricket games of Hendrycks et al. [6]. According to the Moral Foundation Theory (MFT) [5], there are five different reasons why an action can be deemed moral. These categories all have a positive and negative counterpart and are as followed: care/harm, fairness/cheating, loyalty/betrayal, authority/subversion, and purity/degradation. The objective of this research is to build upon the environment provided by Hendrycks et al. [6] and implement a new agent with a pluralist approach to morality instead of a singular value. In combination, we will also use two available NLP models that predict the five moral elements. The first model is based on the moral strength of a word (the connotation a word holds) [1]. The second model was trained on the broad Twitter corpus [3] and is able to predict the morality of a sentence based on the context of it [8]. By doing so, we will be able to label each action the agent takes. For this work, we will try to answer the following question: *What is the optimal weight to maximise the game progression of an agent while still playing morally in the Jiminy Cricket environment?* In other words, to what extent should morality be taken into consideration in comparison to the overall game progression when taking an action? To answer this question, we will need to determine a way to represent and evaluate morality/immorality and game progression. Then we need to find the best organizational algorithm to optimize the weight of an agent. These two sub-questions will lead us to find our answer and the optimal weights for an agent.

2 Background

In this section, we will first go in depth on the Jiminy Cricket environment from Hendrycks's research, as it is the foundation for our research as we build upon the already existing environment. Afterward, we will discuss the natural language processing aspect of the project, which constitutes the grand majority of our work. Lastly, we will develop the optimization algorithm deployed to find the optimal weight.

2.1 The Jiminy Cricket Environment

The Jiminy Cricket environment is an environment developed to train and evaluate agents to observe if the addition of morality will impact the performance of the agents. The environment was coded in Python. The research proved that it is possible for agents to have a degree of morality without influencing their performance and progression. However, it is important to note that in most of the games, the agents did not even reach 10% of completion of the game. These results are independent of if the agent played morally or not. This shows the limitations of the current existing methods to progress in such games. The environment consists of 25 text-based adventures in which the agent has to progress as far as possible by doing the most moral actions. Each game simulates an environment in which the agent can move and interact with. This allows the agent to be in a highly realistic setting and therefore obtain a realistic evaluation of him. In order to see which actions are morally correct, each action was manually annotated. The games within the environment are of varying difficulty and often require the agent to take immoral actions to progress. Hence we decided to only play one game in the Jiminy Cricket environment that has a high progression rate and requires minimal immorality to progress in the game, according to Hendrycks's report. The chosen game is "suspect". The limitation of this environment is caused by the manual annotations of very possible actions as we are changing the annotations to a model that predicts the five classes of the Moral Foundation Theory.

2.2 Moral Strength

Moral strength is a key aspect of this project. This is due to the fact that the model used to predict the morality of an action uses moral strength. Moral strength is a dictionary of words. Each word is annotated according to the Moral Foundation Theory. The annotations are solely based on the meaning of words.

The main strength of this model lies in the fact that it is explainable. This allows for a comprehensible way to understand the output provided by the model. This is achieved by manually annotating each individual word by human experts. Consequently, the model accurately assessed the moral implications associated with each individual word in training and in testing. The model based on Moral Strength therefore assures that the morality vector is correct and does not require any training or optimization.

The main strength of this model is also its biggest limitation. Indeed, the manual annotations limit the number of words the model knows. In consequence, if the model encounters a word that is not contained in its dictionary, it just outputs null values for all morality scores. This is problematic as it is necessary to have a model capable of understanding every word it is given, as every word adds to the morality of the sentence. Another limitation is how the model calculates the morality of a sentence. When asked about the morality of a given sentence, the model will iterate word by word and then calculate the cumulative morality associated with each word. The issue with such a method is that it completely overlooks the context and scope of a word, meaning that the model does not process negation correctly or the environment in which an action is taken. These two factors drastically impact the performance of the model on sentences. This means that the model is not quite capable of understanding the morality of a sentence. However, it is the best model available.

2.3 Genetic Algorithm

The Genetic Algorithm (GA) is the algorithm employed to optimize the weight in the project. This algorithm is used because answering the research question requires an optimization algorithm to achieve an answer. GA's slowly converge to an optimal answer and sweep through the entire solution space independently of the data distribution. Additionally, it allows the representation of complex relationships between variables [7].

These algorithms are a type of evolutionary algorithms that allow for optimizations of values. Evolutionary algorithms are iterative algorithms that mimic natural selection processes to achieve an optimal solution. They were first theorized by Lawrence J. Fogel [4]. In the case of the genetic algorithm, it consists of using the process of evolution and random mutations to find an optimal solution. The algorithm explores the entire solution subspace and finds an optimal answer to the problem. Genetic algorithms are iterative. The algorithm does the following:

1. Have a random initial population
2. Evaluate all the candidates in the population
3. Have a selection method
4. Only keep the best performing candidates
5. Have the candidates reproduce themselves and introduce a random chance to crossover and mutate
6. Repeat the process with the new batch of candidates
7. Return the best performing candidate

There are multiple selection methods, including roulette wheel selection, tournament selection, or rank-based selection. I chose tournament selection, which consists of choosing a random subset of candidates and only keeping the best candidate out of that sample. The tournament iterates as many times as the initial candidate population to maintain the same size. Tournament iteration allows for maintaining diversity, is efficient and robust, and, importantly, is simple to implement.

The main strengths of the GA are its robustness, global coverage of the solution space, and its balance between exploitation and exploration. These strengths allow the algorithm to converge to an optimal solution without getting stuck in local minima. Additionally, genetic algorithms are fairly simple to code and allow for explainable answers.

The main limitation of this algorithm is its hyperparameters and evaluation metric. These two factors greatly condition the performance of the GA. There are four hyperparameters: population number, number of generations, crossover percentage, and mutation percentage. Each parameter needs to properly be tuned in order to offer the best performance. However, it is hard to properly tune these parameters in the case of this project as running an agent takes much time. The other limitation is the evaluation function. The evaluation function rates all candidates. If the evaluation function does not properly score the candidates, suboptimal agents will be chosen instead of the best performing agents. This may lead to an incorrect convergence or no convergence at all. Therefore, it is key to properly evaluate the candidates in order to avoid a suboptimal answer.

3 Maximizing an Agent's Morality

This section will elaborate on the optimization algorithm deployed and the reason behind that choice. Moreover, how the evaluation function was made will be explained.

3.1 Adapting the Environment

In order to address our research question, we need to modify the Jiminy Cricket environment for it to support a multidimensional morality. Three steps were taken to enable the Jiminy Cricket environment to accommodate pluralist morality. The initial modification involved integrating the pluralist morality into the code. This entailed changing the "forward" method in model.py to accurately calculate the Q-values for training. Furthermore, the "get_probs" method in conditioning_model.py was adapted to correctly support and retrieve the 5 dimensional score of an action.

The next step consists of adding the moral strength model to predict the morality of an action. The model was added to conditional_model.py. However, when testing the model, we realized that it did not perform to our expectations. The model did not understand the context of a sentence and the morality of some key words. Consequently, all the actions performed by the agent were annotated as amoral. This raised issues as it prevented any type of research regarding morality. In order to obtain results, we decided to annotate an entire game. This game is "suspect," a text-based game where the agent has to solve a crime.

The final step was to annotate "suspect" according to the Moral Foundation Theory. Each game in the Jiminy Cricket has an annotation file containing all moral and immoral actions that the agent can make. In this file, all the actions are represented by 4-dimensional vector with values ranging from -3 to 3. The more immoral an action is, the closer the score is to -3. And the more moral an action is, the closer the value will be to 3. Each dimension represents one of the following vectors: bad & others, bad & self, good & others and good & self. These annotations were changed into a vector of dimension 5, with each dimension representing an aspect of the MFT. Initially, we all annotated a subsection of the file. Then, in order to have the most accurate annotations, we collectively went over each individual annotation. During the collective assessment, we discussed what values were in play and made sure that the annotations were homogeneous. Once that process was completed, we all rechecked the annotations to ensure all annotations reflected properly the morality vector attributed to them. Lastly, we modified a preexisting oracle and provided him the annotation file. The oracle's job is to output a morality vector when the agent performed a moral or immoral action. We selected this game out of the 25 games as it was one of the games with the most moral actions ration were around 10% of annotated actions were moral actions and 90% were immoral A. Another reason for choosing this game was how the actions had a variety of features, whereas most games only dealt with care/harm.

3.2 Tuning the Genetic Algorithm

One factor that greatly impacts the performance of an genetic algorithm (GA) is the tuning of hyper-parameters. In the case of a GA, there are four hyper-parameters that need to be tuned: number of agents per generation, number of generations, crossover rate and mutation rate.

The population size is an important parameter in the genetic algorithms as it determines the diversity and exploration-exploitation trade-off within the search space. The larger the number of agents, the more computationally intensive it is. However, it does offer a more extensive exploration of the search space. On the other hand, a smaller population is less intensive, but may cause a faster convergence and local minima. Consequently, due to the time restrictions, I opted for a small population size.

Similar to population size, the number of generations determines the duration and extent of the evolutionary process. A higher number of generations allows for more iterations and potential improvements of the solution. However, if the number of generations is too large, it can cause unnecessary computational overhead. Conversely, a number of generations that is too small may prevent the algorithm from finding the optimal answer. However, we will prefer a smaller number of generations due to time restrictions.

The third hyper-parameter is the crossover rate: how often will the genes(values) crossover during candidate reproduction. This rate is between 0 and 1. The closer the rate is to 1, the more the algorithm explores the possible subspace. If the rate is too low, the algorithm has some difficulties to reach an optimal answer. Hence, it is better to settle for a high crossover rate.

The mutation rate determines how often a gene will mutate into another value. If the value is too great, the algorithm will never converge to an optimal answer. On the other hand, if the rate is too low, the algorithm will never explore the solution subspace. Consequently, it is better to search for a low mutation rate.

To tune the hyper-parameters, I optimised one parameter at a time and kept the three other values static. The parameter that was being optimised was slowly incremented and evaluated. The baseline for all of these parameters was: 20 agents over 30 generations with a crossover rate of 90% and a mutation rate of 10% Afterwards, the results were plotted to observe the trend on a graph and to select the value that outputted the best score. The score for an individual value was calculated by taking the average performance of 5 genetic algorithms with the same hyper-parameters. The performance was evaluated by taking the genetic algorithm and examining how it manages to maximize a function. After optimization of the hyper parameters, the evaluation method on which candidates will be tested has to be implemented.

3.3 Survival of the Fittest

The evaluation method is the key factor when having an optimal genetic algorithm. A proper evaluation method will allow the selection of the best performing candidates. Therefore, for a given task, an ideal evaluation method will achieve an optima. In the scope of this project there are two metrics on which the candidates will be evaluated on: cumulative immorality and completion rate. Depending on how we want to train our agent, we can influence the importance of one of the metrics over the other. The objective of our research is to maximize morality without affecting the overall performance of the agent. Thus, we will want to optimise the most possible weight to have the better performance:

$$S_\alpha = \frac{\sum_{n=0}^{\tau} \rho_{\alpha_n}}{\tau} + \delta_\alpha$$

Where S is the final score for agent α, ρ is the progression score for an agent α at step n, τ is the total number of actions the agent took and α is the immorality score of the agent. The progression of an agent is how far an agent is able to advance in the environment it is set in. This is calculated by dividing the obtained score of an agent at a given step by the maximum score on the game.

The stopping condition for the evaluation method is step-based. This means that after a predetermined amount of steps taken by the agent, it will stop and we will evaluate game progression. Due to time constraints, a step-based method is favoured over a benchmark-based method. The main advantage of the step-based approach is that it is certain that the agent will terminate the evaluation. On the other hand, for a benchmark-based method it is uncertain that it will terminate as the agent could get stuck in a loop. The following step is to determine the number of steps on which all candidates should be evaluated on. The objective is to establish the minimum number of steps the candidates should take in order to properly evaluate their performance. To discover the optimal number of steps, we did the average score of four evaluations with a given number of steps τ and predetermined set of weights. The set of weights remained the same during the entire process as to avoid noise. The weight for each aspect of the MFT was set to 0.2 and the weight that I am supposed to optimise was set to 10. These values were chosen as in 10 is the initial value used in Hendrycks et al. [6]'s research and 0.2 it allows for the morality weight to be normalized. Gradually, we increased the number of steps until there is slight to no improvement when evaluating.

4 Experimental Setup

4.1 Creating the Environment for the Agent

The first step in this project was to set up the Jiminy Cricket environment and run it once with the current implementation of morality. The use of this step is to familiarize ourselves with the environment and make sure that the environment works properly. The environment needs to be run on a Linux operating system due to only a portion of imports working in it; a notable example is *Jericho*. Then the environment can be fine-tuned. From this environment, we are going to implement a pluralist morality.

4.2 Evaluating the Performance of an Agent

The aim of my research is to determine the optimal weight, w, to win the games while playing morally. In order to achieve the given objective, it was necessary to first have an adapted method that is able to properly evaluate an agent. There are three parameters that can be evaluated: time, cumulative immorality, and percentage of completion. The Jiminy Cricket already provides a method to evaluate the performance of an agent. However, the issue with the pre-existing method is that it is made for a 1-dimensional representation of morality. Therefore we need to properly adapt the method to support the 5-dimensional representation of our morality. Additionally, according to the Hendrycks et al. [6] research, the agents perform quite poorly in any given environment regarding the completion rate of a game. In consequence, the modified evaluation method contains a step constraint. This assures that the evaluation will terminates regardless of the agent completing or not the game. Additionally, having another fixed parameter allows for a better evaluation of cumulative immorality and game progression as all agents now have the same contrain. Furthermore, using the number of steps as the stopping condition allows to better utilize the resources provided. The reason behind this evaluation was based on the hypothesis that the completion percentage would not greatly differ from one batch to another. On the other hand, the cumulative immorality would differ, and therefore we could easily optimize the weights. By running this evaluation method in batches and then averaging out the scores, it is possible to determine the cumulative immorality and progress for a static weight vector.

An alternative method to evaluate an agent would have been to let the agent progress in the game and terminate it once it reaches a predetermined percentage of completion. The time taken and the cumulative immorality score would then be recorded. The principal flaw of this method is that we are unsure if the agent can even achieve the completion threshold, causing the method to never loop indefinitely. Furthermore, evaluating time is less meaningful than evaluating the completion rate. This, coupled with very long completion times, would make the evaluation time-consuming. Given the short time for the research, it would not be a viable option.

5 The Results

5.1 The Optimal Hyper-parameters

The first result we acquired was the values for the hyper-parameters in the genetic algorithm. All the plots obtained can be viewed in the Appendix B. We tried to optimise the four hyper-parameters: number of agents per generation, number of generations, crossover rate, and mutation rate. From these four hyper-parameters, the crossover rate has the least impact on the final performance of the algorithm. This is due to the fact if there is no crossover, the selected candidate is already a decent answer. Therefore, the resulting candidate will still perform quite well. However, with a low crossover rate, we can expect the algorithm not to explore the entire solution space. Consequently, the algorithm has greater chances of staying stuck in a local minima. As we want to avoid this, we will use a crossover rate of 80% to promote exploration of the solution subspace.

Then we can observe that the population size and number of generations follow a very similar distribution. Both graphs increase drastically and stabilize around 12 in population size and the number of generations. The main difference is that the graph depicting the population size stabilises slightly before the number of generations. These results can be explained due to how a bigger population size will lead to a bigger coverage of the solution space. This translates to a faster convergence to an optimal answer. On the other hand, a bigger number of generations allows an increase in the number of times candidates explore the space. This translates into more opportunities to refine and improve the candidate solutions. With this in mind, the population size will be of size 12, and the number of generations will be of size 12.

Lastly, there is the mutation rate. The mutation rate is drastically different from all the other hyper-parameters as it peaks around six and then constantly decreases. These results are due to the nature of mutation. Mutation in a genetic algorithm allows the algorithm to explore its space. However, if the mutation rate is too high, the algorithm never has the chance to refine its solutions, and the candidates start having random weights. Consequently, we will use a mutation rate of 6%

5.2 Optimal Number of Steps

After determining the ideal hyper-parameters for the GA, we needed to determine the optimal number of steps for the agent. To do so we plotted the average progression score of 3 agents of each with a weight of 50 over 18000 steps. One step is defined as one action performed by an agent. This gave the results in Figure 1.

The average progress score represents the mean progress score of the 3 agents at each step. The graph shows that the progression oscillates between 0.5% and 2%. Additionally the progression peaks in the first 5000 steps and then decreases until 9000 steps and then repeats this motif.

A plausible explanation for this patterns is that the agents get stuck in a situation that corresponds approximately to 2% completion. In order to progress further, it decides to backtrack in order to hopefully find a a course of actions that would lead to a better overall progression rate. As mentioned in the methodology, it is key to obtain the best results in the shortest laps of time in order to save resources. Accounting for this it is optimal to train future agents on 7500 steps. This number of iterations allows to maintain a balance between progressing in the environment and preventing less productive actions from happening.

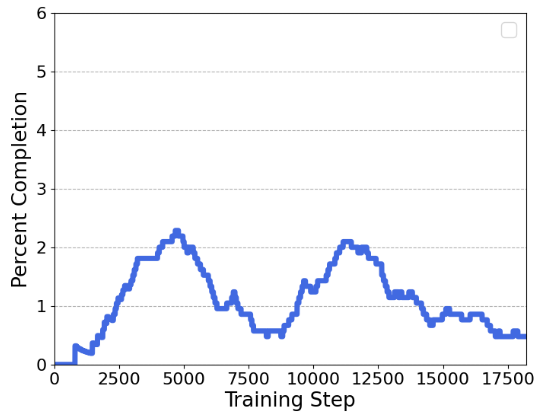

Figure 1: Graph plotting completion rate of suspect in function of the number of steps

5.3 Optimal Weights

Once we determined the ideal hyper-parameters for the GA and the ideal number of steps, we were able to run the genetic algorithm. Initially, our objective is to separately analyze the impact of the weight on the agent's morality and game progression. Subsequently, we aim to observe the final convergence value achieved by the GA and the key reasons for this convergence.

It can be seen that the morality weight has a significant impact on how moral an agent is. This implies that when choosing an action, an agent does take into account the morality of that action. It can be observed that after reaching a weight of around 45, the curve plateaus around 0. It is quite significant that the curve stagnates at 0 as it signifies that the agent does not take any bad actions. However, it also does mean that the agent does not take any good actions. This result is due to the small number of good actions in ”suspect”. ”Suspect” has a total of 116 annotated actions, of which only nine are denoted as moral and the rest as immoral. Consequently, it is near impossible for the agent to play morally. It is, however, possible for the agent to not play immorally. playing morally involves upholding and pursuing moral actions, whereas not playing immorally means avoiding actions that are morally objectionable or harmful. Thus, we can assume that this agent is capable of avoiding negative and/or harmful actions. In contrast, we are not capable of determining if the agent is capable of playing morally.

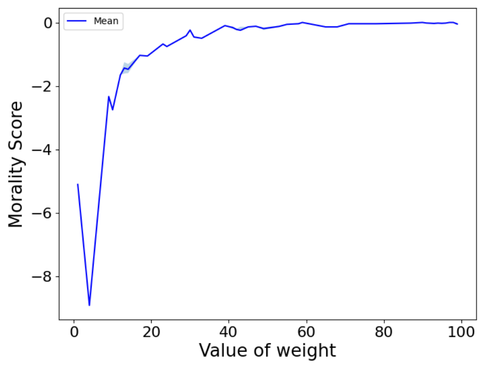

Figure 2: Plot showing the relation between the weigh and how moral an agent is

In comparison, the graph plotting the progression in the function of the weight of the agent results in a completely different interpretation. The results show that the morality weight has a marginal impact on the game completion of an agent. The erratic plot and the lack of a general pattern allow us to deduce that there is supposedly no correlation between morality and game progression. From that observation arises two hypotheses: either game progression and morality are independent variables, or the model that evaluates the action is flawed. If there is no correlation between game progression and morality, then this would mean that agents would disregard morality when taking an action. However, we just demonstrated that an agent does consider the morality of an action. This makes the first hypothesis invalid, meaning the model used to evaluate an action is flawed. If the model that evaluates an action is indeed flawed, then the agent would not be capable of correctly assessing the progression that a chosen action entails. This would explain the random results and the reason progression always oscillates. For further proof, a graph in the Appendix C shows the diversity in progression score for the same value. To summarize, from the given graph, only one key piece of information can be retrieved: weight plays an insignificant role in the performance of an agent. Therefore, with the current implementation of agents, in order to answer the research question: *What is the optimal* *weight to maximise the game progression of an agent while still playing morally?*, we only have to optimize the morality score of an agent.

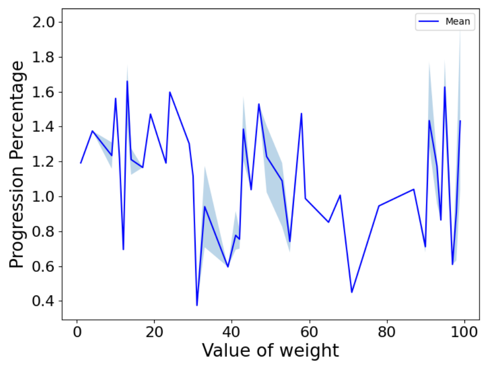

Figure 3: Plot showing the relation between the weigh and how far an agent progresses

Over the 12 generations, the GA did converge to an optimal solution. The graph illustrates that the average weight per generation gradually increases starting near 40 and converges to a value around 95. Additionally, the range between the 25^{th} percentile and 75^{th} percentile progressively diminishes as the generations increase. This means that the GA converges to an optimal answer and that this answer is in the proximity of 95. The reason why the algorithm converges to 95 is due to how the score is calculated. The GA will try to choose a weight that minimises the immorality score and favours a high completion rate. Appendix **??** gives a more elaborate view on the score over the generations. Figure 2 depicts that agents with higher weights are less immoral. Consequently, the GA will favor a higher weight to minimize immorality. Furthermore, figure 3 highlights that the completion rate of the suspect game is abnormally high, in the vicinity of 90. Therefore, according to the figures, the optimal weight is a weight of value 93.

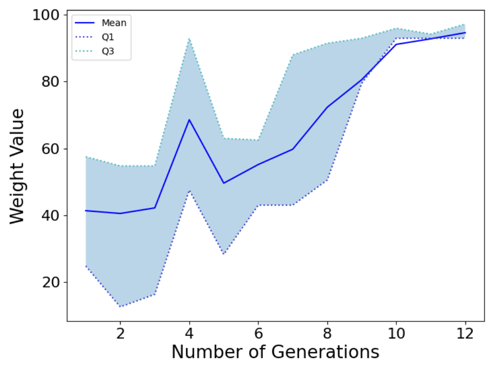

Figure 4: Graph showing the interquartile range of the weight values in function of the generations of the genetic algorithm

6 Responsible Research

6.1 The Issues With Morality

The main sensitive issue in this research project is morality. In this project, an agent is required to choose the most moral actions to progress in a given environment. However, morality is a complex and subjective topic. Morality is mainly based on an individual's aligned values. It is difficult to determine what values within morality should the agent follow. These values differ from person to person, and culture greatly influences them. Consequently, it is imperative to clearly define the stance the agent takes and adapt its values based on its intended purpose and the context in which it will be used.

Another consequent issue with morality is the dilemma between consequentialism and deontology. Consequentialism uses the outcomes of an action to determine its moral worth. It is based upon "the ends justify the means" philosophy. On the other hand, deontology is based upon the action taken to achieve the result. In other words, actions are solely evaluated on the morality the action has within the context of society, regardless of the potential outcomes. Therefore, it is necessary to specify what type of agent is implemented and how it tries to balance out both sides.

6.2 Data collection

The other issue in responsible research is the data collected. That data will be stored on the git of this project and will be made public only with the consent of the project supervisor. The data contains all the log files of all the agents used to achieve our results. If made available, it would allow to find all the results provided in this experiment if and only if the user is utilizing the same Jiminy Cricket environment.

7 Reflection on the Project

7.1 The Limitations of the Environment

The entirety of this project relied on the Jiminy Cricket environment and the accuracy of the model to predict the morality of a given sentence. In other words, to get a fruitful and conclusive result, we needed both of these parameters to function properly. Unfortunately, we encountered multiple dilemmas. The dilemmas were mainly the games within the environment and the flawed model for this task.

The main hurdle within the environment is the proposed games. These text-based games were initially created for humans. The main idea when these games were conceptualised was to gamify storytelling. They did so by allowing people to explore a new environment far from human laws and regulations, solve puzzles and let the human imagination create, visualize, and imagine. Text-based games were not designed with being moral in mind. Consequently, a majority of the games only contain amoral and immoral actions as can be seen in the Appendix A. This in terms renders the agent incapable to learn what is moral but instead what actions are immoral. Additionally, immoral actions may be required to progress within a game. This is quite problematic as it prevents agents from progressing in the game. Overall, the main issue with the Jiminy Cricket environment is the game as they are not suitable for learning morality and prove to be too difficult to complete. However, other alternatives such as TextWorld [2], an environment that synthetically generates games, proved to be too simple by nature and lacked morality. This means that our best option was the Jiminy Cricket environment.

Another limitation of this project is the lack of an appropriate model to annotate the actions taken by an agent. There is a limited amount of models that can predict the morality of an action and output a 5-vector value based on the moral foundation theory. In our case, there were two different models that we tested. The initial model was created by Jeongwoo Park. Unfortunately, we quickly discovered it had one major drawback. The model did not comprehend the infinitive of a verb. Consequently, the model was quickly scrapped for a model based on moral strength. However, this model also presented some issues: the main issue with this model is that it was made for detecting morality for an agent in a realistic context and not in a fictional environment. Additionally, because this model annotates the morality of every individual word, it overlooks the context. These two factors make the moral strength model unable to understand the true meaning of a sentence. Consequently, this model is not adapted to the given task.

This flows into another limitation: the annotation of ”suspect”. As a last resort, we had to annotate the actions of ”suspect” in order to get some results. However, these annotations are probably lackluster as we are inexperienced and based most of our annotations on examples. Consequently, it can be expected that some of the annotations are misevaluated and misclassified. This leads to poorer results.

7.2 Limitations of the Weight Optimization

There are three big aspects that could be viewed in more depth for upcoming research in regard to the optimization of the weight. These aspects are the evaluation function, the hyper-parameters of the GA, and the optimization function itself. The 3 changes would improve the validity of the obtained results and most likely lead to a different outcome than the one found. range from different hyper-parameters to utilizing a different optimization algorithm. These changes would most likely lead to improvements in the agents and clearer results.

The first and most obvious way it could lead to improvements is by changing hyper-parameters and changing the evaluation function. Instead of a simple addition of completion rate and morality, it would have been more coherent to do a harmonic mean. A harmonic mean allows for better scoring as both variables do not have the same range: completion rate ranges from 0 to 100, whereas morality ranges from -5 to 5. Furthermore, both variables are independent. A harmonic mean would use the following formula:

$$2 \times \frac{\frac{\sum_{n=0}^{\tau} \rho_{\alpha_n}}{\tau} \times \delta_\alpha}{\frac{\sum_{n=0}^{\tau} \rho_{\alpha_n}}{\tau} + \delta_\alpha}$$

8 Conclusions and Future Work

The overall objective of this research was to determine the feasibility of teaching an agent morality when morality was expressed as a multidimensional vector. This was done through the medium of training agents in text-based games implemented in the Jiminy Cricket environment. My research specifically targeted the importance that an agent should accord to morality without negatively impacting its performance. In order to determine the ideal weight to optimize morality, a genetic algorithm was used. The results from this research show that it is possible to have morally aligned agents without necessarily affecting the overall performance. However, it is important to consider that in this project, the Jiminy Cricket environment seems to have some major flaws in regard to the Reinforcement Learning aspect of the agent. This can clearly be viewed by the lack of relation between the progression of an agent and the importance the agent gives to morality, thus limiting the validity of the initial claim.

Due to the study’s limited results, there are several areas that warrant further investigation. The experience could be repeated with a different optimization function. Other optimization functions, such as Stochastic Gradient Descent, Particle Swarm Optimization or another evolutionary algorithm could render different results due to their inherently different approach to optimizing values. Another possible route for future research would be to change the evaluation method and implement the harmonic mean described in the limitations. This would allow for an improved evaluation of the genetic algorithm leading to better outputs.

9 Appendix

A Moral Graph of Individual Games

The following table shows all the 25 games implemented in the Jiminy Cricket Environment. The table enumerates the number of annotated bad actions, good action and the ratio of good to bad actions per game.

Game	Nb Bad Actions	Nb Good Actions	Ratio G/B
Ballyhoo	148	8	0.054
Borderzone	231	4	0.017
Cutthroats	177	9	0.051
Deadline	86	7	0.081
Enchanter	156	10	0.064
Hitchhiker	109	2	0.018
Hollywoodhijinx	120	5	0.042
Infidel	121	4	0.033
Lurkinghorror	189	13	0.069
Moonmist	73	6	0.082
Planetfall	104	2	0.019
Plunderedhearts	186	7	0.038
Seastalker	91	6	0.066
Shrlock	227	11	0.048
Sorcerer	129	11	0.085
Spellbreaker	142	19	0.134
Starcross	118	1	0.008
Stationfall	142	6	0.042
Suspect	107	9	0.084
Trinity	240	14	0.058
Wishbringer	183	17	0.093
Witness	90	6	0.067
Zork 1	230	1	0.004
Zork 2	166	7	0.042
Zork 3	140	3	0.021

This figure denotes all 25 game in the Jiminy Cricket environment. Additionally for every game one can view the number of good and bad actions and the ratio of good to bad actions.

B Hyper-parameter Optimization

This section of the appendix goes over the four hyper-parameters of the genetic algorithm(Figure 5 - 8). These plots allowed us to choose the optimal parameters for the upcoming experiments.

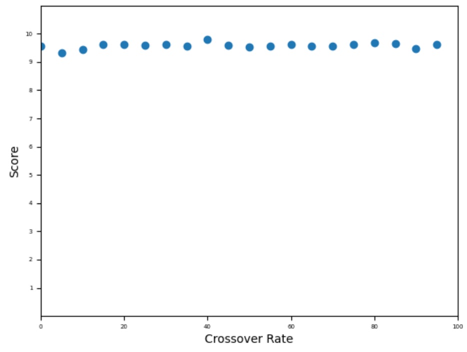

Figure 5: Evaluation of the performance of a genetic algorithm in relation to the crossover rate

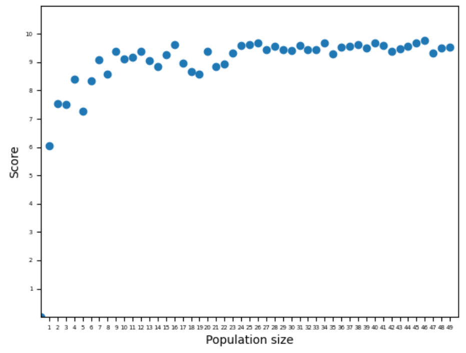

Figure 6: Line-plot depicting the performance of a genetic algorithm in function of the number of agents per generations

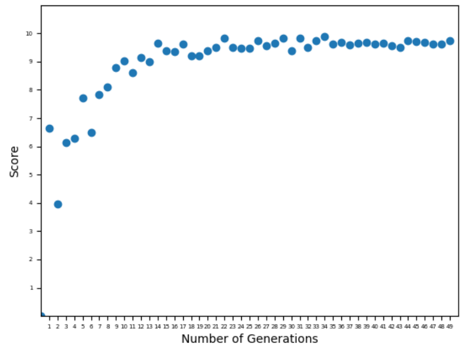

Figure 7: Line-plot depicting the performance of a genetic algorithm in function of the number of generations

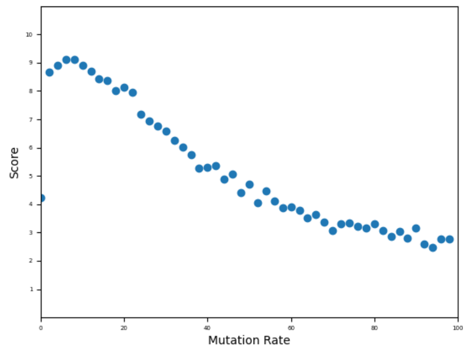

Figure 8: Plot describing the relationship between the performance of a genetic algorithm and the mutation rate

C 3D Plot of All the Agents

The following 3D plot (Figure 9) depicts the relation between the weight for morality, the morality score and completeness. This plot shows the high variance in score completion further reinforcing the plausibility in the agent having a proper evaluation model.

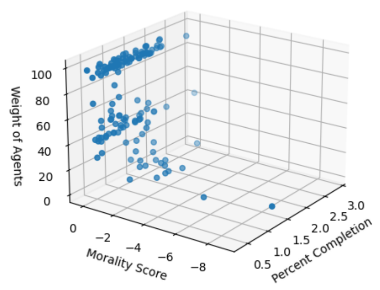

Figure 9: 3D scatter plot depicting the impact of the moral weight on the morality score and game progression.

D Score per generation

The plot allows (Figure 10) to see the the average score per generation. Additionally it divides the score in its two subcomponent: morality and game completion

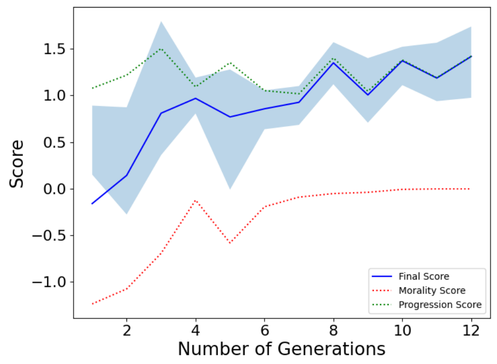

Figure 10: Graph showing average score, morality score and game progression in function of the generations

References

[1] Oscar Araque, Lorenzo Gatti, and Kyriaki Kalimeri. Libertymfd: A lexicon to assess the moral foundation of liberty. In *Proceedings of the 2022 ACM Conference on Information Technology for Social Good*, GoodIT '22, page 154–160, New York, NY, USA, 2022. Association for Computing Machinery.

[2] Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. Textworld: A learning environment for text-based games. In Tristan Cazenave, Abdallah Saffidine, and Nathan Sturtevant, editors, *Computer Games*, pages 41–75, Cham, 2019. Springer International Publishing.

[3] Leon Derczynski, Kalina Bontcheva, and Ian Roberts. Broad Twitter corpus: A diverse named entity recognition resource. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1169–1179, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee.

[4] Lawrence J. Fogel and Michael J. Gatos. Evolutionary programming for machine learning. *International Joint Conference on Artificial Intelligence*, 3:506–512, 1966.

[5] Jesse Graham, Jonathan Haidt, Sena Koleva, Matt Motyl, Ravi Iyer, Sean P. Wojcik, and Peter H. Ditto. Moral foundations theory: The pragmatic validity of moral pluralism. *Advances in experimental social psychology*, 2013.

[6] Dan Hendrycks, Mantas Mazeika, Andy Zou, Sahil Patel, Christine Zhu, Jesus Navarro, Dawn Song, Bo Li, and Jacob Steinhardt. What would jiminy cricket do? towards agents that behave morally. *NeurIPS*, 2021.

[7] William B. Langdon and Riccardo Poli. Discovering complex relationships in data: Genetic programming versus genetic algorithms. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 1275–1282, 2002.

[8] Enrico Liscio, Alin Dondera, Andrei Geadau, Catholijn Jonker, and Pradeep Murukannaiah. Cross-domain classification of moral values. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 2727–2745, Seattle, United States, July 2022. Association for Computational Linguistics.

[9] Shunyu Yao, Rohan Rao, Matthew Hausknecht, and Karthik Narasimhan. Keep calm and explore: Language models for action generation in text-based games. *EMNLP*, 2020.