Ai scientific paper

Rules based AI Is a model Which bases itself on a predefined set of if, then rules. The principle of it is to mimic human decision making within a structured, predictable framework. For example, some expert systems were designed to replicate the decision making capabilities of human professionals within a specific discipline, while constantly learning new patterns and expertise. One of the best things about view rule based AI models is the transparency and predictability in the functions. So essentially in a concise manner, Rule based AI provide structured mythology to process data and make decisions with clarity and predictability. (5) by programming a set of rules into Python And making loops to run them I effectively created a rules based AI system in a simple form to help me solve a problem called the prisoner's dilemma.

The person's dilemma Is one of the Main problems in game theory. It was introduced in 1950 by the mathematicians Merrill Flood and Melvin Dresher (3), explores the dynamics of cooperation. And. Defecting. In the context of self – interest and human nature and can be used in situations from criminal to social, including business and international diplomacy, which I cover more in the latter stages of this essay. Because of the implications and modelling of human nature and decision – making, I decided that it would be a good topic to try and make an A I model out of to explore the teacher student relationship. The basis of the problem is set out in a way that both prisoners are taken to separate cells and given the opportunity to tell their stories. No communication is allowed before, during or after between them. If the prisoner ‘defects’, essentially proving the other guilty, provided the other prisoner ‘cooperates’ (does not give any details why the other should be guilty) then they will be freed with no further consequence. However, the other prisoners will be given the maximum time for the crime they have committed and the same works if it were the other way round. If, however, both prisoners decide to ‘defect’ (give each other away as guilty) both will serve an equal and less amount of time, which is similar to if both ‘cooperate’ (do not give each other away) but in this circumstance they would both serve the full sentence. There are also many more ways they could choose to respond in the case that they are accused of more than one crime, for example, open of the prisoners could just decide to ‘hard defect’ meaning that no matter what they will always blame the other prisoner and name them as guilty. In this context the prisoner that chooses to hard defect will either get no sentence or a shortened one as the other prisoner would either ‘defect’ too or ‘cooperate’. In another context, the prisoner could use the ‘tit for tat’ scenario, meaning that if the other prisoner ‘cooperated’ the first time, they would ‘cooperate the second time but if the other prisoner decided to ‘defect’ they would then defect.

This algorithm scenario will work in the context of my problem of the teacher – student relationship using the same ‘cooperate’, ‘defect, ‘hard defect’ and ‘tit for tat’ processes of social interaction. The real – life interactions would include the following situations; a teacher who cares about the student's work and well-being, and a student who reciprocates. While both parties are reciprocal, the social interactions will be ‘cooperative’. If the student decides to ‘defect’, in the instance the teacher follows the ’tit for tat’, the teacher will also start to ‘defect’ but if the student later then decides to ‘cooperate’ again, the teacher will follow. In the instance the teacher uses the ‘hard defect’ method, the teacher will continue to ‘defect’ once the student has, even if they end up ‘cooperating’ in the end. Similarly, if the teacher were to have the same interactions, the student would follow suit with either ‘tit for tat’, ‘hard defect’, ‘cooperate’ or ‘defect’. Whilst exploring these mannerisms through creating a rule-based AI module, the processes will be tested and analysed against a ‘random’ series of interactions for effective comparison and having a fair test.

Whilst researching ‘the prisoners Dilemma’ I came across a variety of sources which effectively test and explain the model in situations relevant to the world, for example it is said that “it is incorporated in every disciplined social act, for instance in an orderly evacuation of a burning theatre” (1) or “used by political scientists to describe interactions between international actors from the individual decision making of Thucydides during the Peloponnesian War, as outlined in the Melian dialogue, to events during the Cold War period(s) modelling conflict and negotiation between the United States of America and the Soviet Union .” (2) During the Cold War, the Prisoner's Dilemma was particularly relevant in modelling the interactions between the US and the Soviet Union. The decisions around nuclear arms, where each side had to decide whether to cooperate by limiting their arsenals or to defect by continuing to build up weapons, mirrored the dilemma. The Cuban Missile Crisis is one key example where President John F. Kennedy and Soviet Premier Nikita Khrushchev had to navigate a highly dangerous standoff, making decisions that could either lead to mutual destruction or de-escalation. In this scenario, both leaders ultimately chose to cooperate, leading to a resolution of the crisis. This historical context shows how the Prisoner's Dilemma can be applied to real-world scenarios involving critical decisions between major powers. (4) While these are both true and backed-up statements, I wanted to explore a situation that is applicable to people of similar age and experience to mine for the purpose of relatability, which is not true for the thesis giving the example of political scientists and world leaders. Taking the relationship between a teacher and student is easy to model due to the nature of the program, which follows the same pattern and has been experienced by many of us at various ages throughout our lives. As well as this, it explores the notion of trust, which is the foundation of meaningful relationships, especially in educational settings. Trust fosters communication, mutual respect, and engagement throughout the learning process, benefiting both parties. By exploring the teacher-student relationship in terms of trust, this algorithm can massively help in understanding how trust in social situations can be broken, made, or even strengthened.

To test my experiment, I created algorithms for each of ‘tit for tat’, ‘hard defect’, ‘defect’, ‘cooperate’ and ‘random’. I then created loops for each to run 100 times to create a data set that could be used to compare against other studies and the individual cases. For ease of analysis of data, I used a points scoring system which allowed me to create winners and ranked places. The results were as follows:

Random: Wins = 900, Score = 195918

Defector: Wins = 500, Score = 95084

Cooperator: Wins = 0, Score = 37554

Tit-for-tat: Wins = 0, Score = 37488

Upon first glance, it seems that to defect would be in the best interests of social situations as it won with 500 wins and 95084 points compared to the cooperator with 0 wins and 37554 points and the tit for tat again with 0 wins and only 37488 points. This could be true in some contexts, but in the real world, that behaviour would lead to a very unproductive society. Additionally, the defectors would be in constant competition with each other. In the context of the student-teacher relationship, it would make the school/education environment a very unpleasant place to be, and no learning would be achieved, as constant defection would mean the teacher would not teach or engage and consequently the student would not learn or engage. The tit for tat could also be a long slope, leading to the erosion and breakdown of relationships as minor offences lead to escalation through retaliation, which ultimately goes against one of the main structures of society itself: collaboration, which is seen as one of the most important attributes and central to success. The fact that the cooperator wins over the tit-for-tat makes for an opportunity to deeper understand the complexity of social situations and the very nature of the prisoner's dilemma. In the context of the teacher-student relationship, it is not surprising that the cooperator algorithm would win, as the situation where the teacher and students have a good working relationship is the most ideal. The cooperator algorithm would likely build trust between students and the teacher, as well as understanding and long-term success in social situations. But there is no opportunity to gain the highest individual attributes. Cooperation would also create a productive environment unlike defection and provide the right situations for constructive feedback, which is the backbone of education, no matter the level. The fact that cooperation did not win over tit for tat shows that as a society, even though being modelled as artificial intelligence, we value personal gain over the overall environment.

In conclusion to my experiment, which was based on the prisoner's dilemma in the context of a teacher-student relationship, provides insights into social situations and human behaviour. Several decision-making techniques such as defect, cooperate, tit for tat, and hard defect make different impacts on society and, typically in my case, the teacher-student relationship, and the environments this creates: either productive, unproductive, or uncooperative. Expanding on this, the defector strategy delivered the highest individual score, implying that for society to succeed, that would be the ideal way that everyone should react in social situations. However, it only works in the interest of the person who is defecting, and if everyone defected, it would cause chaos and long-term issues, especially with the teacher and student, as the teacher would not make any progress in their career, and the student would not make any progress in their education, hence both not having a good future set up, and in turn, hurting society overall. Seeing cooperation at the highest score should mean that is the best way to act in social situations. However, in real life, this does not always work as it does not create a high yield of personal gain, only benefits a group, and can sometimes be the opposite of what the individual wants to do. In a teacher-student context, this would also be the ideal way to go, as the teacher would often prioritize the student, help them, provide marks and feedback, and overall create a great learning environment for the student, and the student would reciprocate, creating a powerful and positive working environment for the teacher, and also benefit themselves by getting good exam grades and setting themselves up for a great future. But in the real world, students and teachers are not perfect, and they often have days where they do not want to cooperate, and so tit for tat becomes relevant with the highest score, as it seems to be the most likely and reasonable interaction in social situations and between teacher and student, where if the teacher is cooperating, the student will also cooperate, but if the teacher or the student defects, the other will do the same. This is more in line with human nature and can be useful to create competition and incentives for people who want to do well, but will very quickly leave behind those who do not. This is because the tit for tat social interaction shows the importance of trust in relationships, especially between teacher and student, and shows that while defection may offer short-term rewards, cooperation ultimately leads to success.

In a critical reflection of my work, creating the loops in which the algorithms would run was the most difficult part, this is because not only did I need to do every algorithm 100 times to do a fair test I also had to make sure that I was testing each algorithm individually against its own algorithm in two instances one for the teacher and one for the students. using classes allowed me to effectively code and be able to call them in various places without repeating code too many times which helps with having to do multiple runs of the same algorithm. the AI algorithm I have created may also not be completely accurate this is because there is only so many situations a programme of this level can create and execute as well as the fact that the cooperator and effector algorithms maybe too simplistic for the model of social situations in real life, for example in social situations there's always emotion involved and different levels of cooperation and affection so in order to make this more accurate I would include a percentage of hard defection and hard cooperation which would make it more or less for example annoying or heartwarming in the consideration of the situation and the context of what has happened before and after. This could also just be done by having a more complex rule system. 

#the basis of this code was taken from multiple internet searches and compiled by chatGPT and then tweaked#

import random

#initialises payers with sore and the type they are playinhg as and with what strategy

class Player:

    def \_\_init\_\_(self, name):

        self.name = name

        self.score = 0

    def reset(self):

        self.score = 0

    def strategy(self, opponent):

        pass

#cooperating algorithm

class Cooperator(Player):

    def \_\_init\_\_(self):

        super().\_\_init\_\_("Cooperator")

    def strategy(self, opponent):

        return 1  # Always cooperate

#defecor algorithm

class Defector(Player):

    def \_\_init\_\_(self):

        super().\_\_init\_\_("Defector")

    def strategy(self, opponent):

        return 0  # Always defect

#titfor tat algorithm

class TitForTat(Player):

    def \_\_init\_\_(self):

        super().\_\_init\_\_("TitForTat")

        self.last\_opponent\_move = 1  # Initially cooperate

#if last move was defect, player will defect

    def strategy(self, opponent):

        return self.last\_opponent\_move

#random control variable to test against

class Random(Player):

    def \_\_init\_\_(self, prob\_cooperate):

        super().\_\_init\_\_("Random")

        self.prob\_cooperate = prob\_cooperate

    def strategy(self, opponent):

        return 1 if random.random() < self.prob\_cooperate else 0

#the game initialised

class Match:

    def \_\_init\_\_(self, player1, player2, turns):

        self.player1 = player1

        self.player2 = player2

        self.turns = turns

    def play(self):

        self.player1.reset()

        self.player2.reset()

        for turn in range(self.turns):

            move1 = self.player1.strategy(self.player2)

            move2 = self.player2.strategy(self.player1)

            # Ccalculates scores, apecific to each variable of the game

            if move1 == 1 and move2 == 1:  # Both cooperate

                self.player1.score += 3

                self.player2.score += 3

            elif move1 == 1 and move2 == 0:  # Player 1 cooperates, Player 2 defects

                self.player1.score += 0

                self.player2.score += 5

            elif move1 == 0 and move2 == 1:  # Player 1 defects, Player 2 cooperates

                self.player1.score += 5

                self.player2.score += 0

            elif move1 == 0 and move2 == 0:  # Both defect

                self.player1.score += 1

                self.player2.score += 1

        return self.player1.score, self.player2.score

#plays the game

class Tournament:

    def \_\_init\_\_(self, players, turns=50, repetitions=100):

        self.players = players

        self.turns = turns

        self.repetitions = repetitions

    def play(self):

        results = {player.name: {'wins': 0, 'score': 0} for player in self.players}

        for \_ in range(self.repetitions):

            for i in range(len(self.players)):

                for j in range(i + 1, len(self.players)):

                    match = Match(self.players[i], self.players[j], self.turns)

                    score1, score2 = match.play()

                    if score1 > score2:

                        results[self.players[i].name]['wins'] += 1

                    elif score2 > score1:

                        results[self.players[j].name]['wins'] += 1

                    results[self.players[i].name]['score'] += score1

                    results[self.players[j].name]['score'] += score2

        return results

#runs it 100 times

def torn(players):

    tournament = Tournament(players, turns=50, repetitions=100)

    results = tournament.play()

    # Print results: Wins and scores for each player

    for player in players:

        print(f"{player.name}: Wins = {results[player.name]['wins']}, Score = {results[player.name]['score']}")

    # Rank players by wins

    ranked\_players = sorted(players, key=lambda p: results[p.name]['wins'], reverse=True)

    print("\nRanked Players by Wins:")

    for rank, player in enumerate(ranked\_players, 1):

        print(f"{rank}. {player.name} - Wins: {results[player.name]['wins']}")

    return results

# Example usage of the functions

players = [

    Random(0.5), Random(0.8), Random(0.2),

    Defector(), Cooperator(), TitForTat()

]

# Call the tournament function

torn(players)

Reference list

1. Rapoport, A. (1974) ‘Prisoner’s dilemma — recollections and observations’, *Game Theory as a Theory of a Conflict Resolution*, pp. 17–34. doi:10.1007/978-94-010-2161-6\_2.
2. Longford, Edward (2022) *Modelling Group Dynamics with SYMLOG and Snowdrift for Intelligent Classroom Environment.* PhD thesis, University of Essex.
3. Team, T.I. (no date) *What is the prisoner’s dilemma and how does it work?*, *Investopedia*. Available at: https://www.investopedia.com/terms/p/prisoners-dilemma.asp (Accessed: 23 November 2024).
4. (No date a) *Encyclopædia Britannica*. Available at: https://www.britannica.com/topic/arms-race/Prisoners-dilemma-models (Accessed: 23 November 2024).
5. Copilotly (no date) *Rule-based AI*, *Copilotly*. Available at: <https://www.copilotly.com/ai-glossary/rule-based-ai> (Accessed: 24 November 2024).