ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DEPARTMENT OF COMPUTER SCIENCE
AND ENGINEERING

ARTIFICIAL INTELLIGENCE

MASTER THESIS

in

Multi-Agent Systems

THE AGENT'S DILEMMA:
A COMPARATIVE STUDY OF
LLM-BASED ARCHITECTURES IN STRATEGIC DECISION-MAKING

CANDIDATE
Davide Cremonini

SUPERVISOR
Prof. Roberta Calegari

CO-SUPERVISORS
Prof. Paolo Torroni
Prof. Christian Arnold

Academic year 2024-2025
Session 5th

Contents

Introduction

Intelligence is one of the defining traits of the human species. While the concept of intelligence itself has been the subject of scientific and philosophical evaluation for millennia, technology kept advancing and is now approaching an increasingly convincing mimicry of human intellectual capabilities. Artificial intelligence, while being a topic of study for the last half century, has recently become an increasingly popular subject of investigation and focus of attention, not just from the computational science community, but from all branches of research. Additionally, aside from its increasing role in academic studies, the diffusion of accessible artificial intelligence technologies to a wider public made them a global phenomenon, the consequences of which are still to be evaluated. In the last years, Large Language Models (LLMs) took the spotlight, becoming for the broader public synonymous with AI. The reason for this is their apparent capability of reproducing human speech patterns and interact in a user-friendly and language-driven way, which made them accessible tools even for other fields of research.

LLMs are deep learning models which, given a set of encoded words (context), produce a new sequence of tokens; ideally the input set is a question or a piece of a conversation, while the output is the appropriate answer or response. This association is achieved through probabilistic inference. As for all machine learning models, the utilized function is not directly specified by the developer; it is instead learned by processing an enormous amount of conversational data. The objective is to make the inference as precise as possible, thus producing an increasingly more convincing response. With these premises, replicating human behaviour through these means may seem plausible, given enough data. Reality, however, seems to contradict this. Recent studies proved that these systems, while good at simulating the general human behaviour, still fall short when it comes to capture the individual nature of a human being: they seem to lack the cognitive capabilities that are commonly associated with humans such as a sense of self or, more fundamentally, inherent reasoning capabilities. This is not surprising, since LLMs act more like "statistical parrots[5]", formulating complex and structured sentences without fully realizing what they are actually saying.

From the point of view of social scientists, LLMs represent an interesting tool to simulate human behaviour, as they would allow bypassing the necessity of gathering a human sample. There are, however, a few issues with this approach, some of which are directly tied to the inherent problems of LLMs discussed above. Among the numerous research institutions that are investigating these applications of LLMs there is the Centre for Artificial Intelligence in Government (CAIG), a recently instituted team working under the School of Government at the University of Birmingham: one of their project involves the development of a simulation of the negotiation process of the European Council. This project would require the construction of a multi-agent system of LLMs, with each agent taking the role of a negotiator tasked with defending its ideals and goals. The project itself is ambitious and bound to clash with the inherent difficulty of reproducing consistently a human personality through a plain LLM. Given these limitations, the CAIG team is now investigating new ways to enhance each agent's cognitive capabilities to shorten the distance between the simulation and reality. This thesis is the result of a collaboration with CAIG's team on the topic of LLM agents enhancement to imbue them with higher cognitive capabilities and influencing their decision-making process. Overall, the dissertation is structured in two main parts, followed by some concluding remarks.

Part I introduces all the theoretical premises that serve as foundation for the project with a broad presentation of the current role of the application potential of AI in negotiations (P. I, Ch. 1). It is followed by an introduction to game theory as a bridge between social-economic sciences and maths (P. I, Ch. 2). On the computational science side, the following chapters will present the current state-of-the-art technologies in the field of LLMs and agent modelling(P. I, Ch. 3) and introduce the experiment of the second part by providing an overview of the related work on the topic(P. I, Ch. 4).

Part II is dedicated to the experiment itself: a comparison between different architectures for agents in the context of a game theoretic experiment: first will be presented a broad description of the framework for the experiments (P. II, Ch. 1). Following it there is a presentation of the three main architectures whose behaviour will be the object of the inquiry (P. II, Ch. 2). Lastly, the results of the experiment will be presented and analysed (P. II, Ch. 3), concluding discussing the results of the experiments and addressing some of their limitations (P. II, Ch. 4).

Part I

Part I — Theoretical Foundations

Premises

One of the greatest achievements of any scientific theory is the definition of a model. A model is a system which, given a set of prerequisite conditions is capable of producing a prediction with a certain degree of accuracy. Modelling a problem can be a complex task which often requires finding a trade-off between reality and manageability. On one side there is the duty to model the subject with as much attention as possible. On the other there is the necessity to avoid a degree of complexity that any potential user of said model may find too challenging. The deciding factor on this difficult balance is the objective of the modeller. If responsiveness is one of the prerogatives, then one may favour a simpler, yet faster model.

This first part of the dissertation is dedicated to modelling problems both from a theoretical and a more practical point of view. As stated before, modelling is a practice embedded in all research fields. This thesis is an attempt at creating a bridge between some of them, showing the potential of artificial intelligence as a cross-domain tool.

Chapter 1

Artificial Intelligence in Negotiation and Diplomacy

1.1 A new age for human negotiations?

Artificial intelligence is changing the landscape of numerous disciplines: tasks previously reserved for humans are being approached by automated systems with increasing frequency, the rise of LLMs and their mass deployment is affecting they way people communicate in everyday life. Given these premises, it is not unfathomable to think that this change is propagating at the highest levels of communication, such as companies and diplomatic negotiations.

Nowadays, artificial intelligence systems have become a widespread tool in the world of diplomacy. There are, nonetheless, a few issues to address, but in the long term, their benefits will provide a great incentive to their adoption. In an increasingly complex world, where historical records and data are more accessible than ever, decision-making can be rethought from being an action guided by the experience of an expert to a data-driven prediction with varying degrees of accuracy. In this context, machine learning models are the perfect tool to reign in this complexity, drawing correlations with a precision no human in the past could achieve. It's not just that, however: diplomacy used to be an art in of itself, comprising a long process of careful planning for the parties involved and a constant refining of the different stances to find an agreement and avoid conflict. Now, thanks to LLMs, some of these processes are starting to be delegated to automated systems, as their ability to smooth over disagreements and mediate becomes more and more reliable. Currently, artificial intelligence is present in the form of tools for informed decision-making, for the simulation of bargaining scenarios or to review propositions. However, the potential for these technologies is limitless as more and more tasks are devolved to them. This evolution seems only natural, never in history a wider source of data has been available for negotiating actors to extract information from: past errors could be avoided and previously unthought solution can be discovered earlier.

While this perspective may seem fascinating and, despite all, worth pursuing, there are some critical shortcomings that can spawn from the overreliance on these systems, most of which come directly from their innate flaws. The most glaring issue lies within the data these systems are trained upon. While historical events and data are accurate, most of them are not exempt from bias. If these systems are supposed to learn from the past, how can we make sure they will not reflect past mistakes or, even worse, suggest options that will amplify them? Another problem is one of accountability: while it is comforting the idea of delegating such complex decisions to automated systems, the field of diplomacy is one where the consequences of a bad decision may have the most devastating implications. Lastly, there is an issue of affordability: while these technologies seem powerful, their availability is not evenly spread across the world. More resourceful countries could take advantage of this disparity to tilt the scale even more on their favour by leveraging their knowledge as a powerful bargaining tool.

Considering all of these premises, it is impossible to not recognize that artificial intelligence is ushering a new era in the field of diplomacy. Fortunately its digital nature makes it possible to gather information on how things are changing, to study these new tools and the impact they are having, to understand them and, ultimately, to learn how to act around them. In this regard, this study constitutes a small tile of a larger mosaic that aims at investigating this phenomenon, helping its development but avoiding its pitfalls.

1.2 Brief introduction to diplomacy theory

In this section, a few useful definitions will be provided as a framing device for this entire thesis in the larger field of AI-assisted diplomacy and negotiations. Diplomacy is a practice as old as human history: when two human groups have conflicting interests, communication between them has always been the best tool to avoid armed confrontation, and whenever that was inevitable, negotiating would become crucial in the aftermath to settle the new status quo. For the sake of this dissertation we will define Diplomacy as *"a process between actors (i.e. diplomats) who exist within a system and engage in a private and public dialogue to pursue their objectives peacefully"*. This definition is agnostic to the context where the interaction is taking place and can therefore refer to any level, from a local council of residence to the highest contacts of foreign policy. Considering this, a negotiation process can be analysed from three main perspectives, each taking into account one separate aspect that can affect the end result of the discussion.

- **Power resources**, the resources each party has at its disposal to push its interests. These can encompass different factors, ranging from military or economic power (on a national level) to contacts and local influence on a smaller scale. While not the only deciding factor, these represent the starting point as each actor knows what they can offer and what others can respond with.

- **Strategy**, this represents how an actor intends to use its resources to reach a certain goal. Each resource can be viewed as an instrument and using them carefully is the key to achieving the desired goal. By reasoning on these, a diplomat has to decide what demands to prioritize, when to cooperate with other actors or when to proceed alone.

- **Personality of the individuals**, this represents the diplomat's personality which, since discussion entirely relies on human interaction, is always a deciding factor. Personal charisma and the choice of words, despite any appeal to objectiveness, affect decision-making and can tip the scale in the direction of one faction or another.

By observing this partition it is easy to see how the introduction of artificial intelligence can influence each one of these aspects. As resources themselves, AI systems are already at the centre of the debate, since their availability as services is a powerful tool in the hands of the providers. On top of this, resource management and planning through automated systems allows for the definition of more elaborate strategies and more accurate assessments before the negotiation even begins. Lastly, with the possibility of language refinement enabled by LLMs, even the most "human" part of diplomacy may be subject to some form of optimization, in an effort to strike better compromises. Having existed for millennia, diplomacy has taken many forms and thus, many types of if were classified throughout history. Here we report a short list of the most notable.

- **Economic**, it is a form of diplomacy that utilizes the economic tools available to push forward a specific agenda. In this case direct coercion is almost never necessary, since its main tool is economical incentives: on a national level this may include trade deals, lendings or, in more serious cases, sanctions and tariffs.

- **Gunboat**, one of the most ancient and direct types, it involves a show of force, usually military, to intimidate the other actor into making concessions out of fear of a direct confrontation.

- **Scientific**, as a consequence of the globalization of the scientific community, this form of diplomacy involves the consolidation of relationships by sharing scientific discoveries, with the main incentive that scientific progress thrives in a larger community.

- **Humanitarian**, this type involves the deployment of resources in situations where human lives are at risk with the objective of preventing harm and establishing a mutual effort to help the actors involved in the future.

Now, having provided a brief introduction to the world of diplomacy, we can follow by presenting some cases which document the current impact of AI technologies in this field.

1.3 AI negotiations, some case studies

The following research studies investigate the potential of application of artificial intelligence systems in different negotiation fields, all tackling current issues.

1.3.1 The Habermas Machine

Identifying political polarization as a key issue of today, the authors introduce the Habermas Machine[24], an artificial intelligence negotiator to mediate the stances inside a divided community. The system itself works by collecting different opinions from members of a community and preparing a final statement with the objective of maximizing total approval. Once the statement is produced, it is possible to critique it for additional refinement. This model was tested on a group of 5700 UK citizens, and tasked with mediating controversial issues. The results showed that most of the sample preferred the statements written by the AI-mediator over those written by human mediators. Under the evaluation of external judges, the machine-produced statements were deemed more clear, informative and less biased. As participants were asked to vote for a statement concerning a particular issue, it was observed that unanimous approval rates increased significantly with AI-written statements. Lastly, the Habermas Machine was also tested on a real-world case of UK residents in a virtual assembly addressing topics such as Brexit, immigration and climate change and the result mirrored those of the controlled experiment. A noteworthy observation by the authors is that the Habermas Machine seems to respect minority opinion, addressing their concerns in the statements and thus including all possible points of view. This experiment seems to suggest a great potential for AI to address one of the key issues of today by smoothing political divisions and helping reaching an agreement in small communities. Nonetheless, some issues still remain unaddressed, such as the ever-present problem of bias and the necessity, for this system to work, that citizens engage with it in good faith.

1.3.2 AI in Peacemaking

This report[18] from the Center for Strategic and International Studies describes how an LLM-based system was consulted to analyse the current state of negotiations concerning the ongoing war in Ukraine. Specifically, the Retrieval Augmented Generation technique was deployed, by providing the agent with a dataset of over 300 historical peace agreements from the last 30 years (i.e. the Uppsala Conflict Data Program). Based on these inputs the system was prompted to describe which were the common negotiation parameters in the corpus, abstracting the most critical aspects from previous deals. After the initial setup, the system was provided with 79 articles which described the current situation and presented the two belligerent faction's demands. Among these there were proposed peace deals offered by the two parts. With this additional information, the system was prompted again, asking which historical patterns it managed to recognize in the current situation and which were the most important aspects to consider in a future peace agreement. According to the system, any potential peace agreement had to take into account three main factors: territorial control and security guarantees; a form of trade-off between international justice and the situation on the field; and the necessity of international involvement. Lastly, the system was prompted to generate three potential peace deals that accounted for the highlighted key issues, which will be used as groundwork for potential proposals. This experiment showed the potential of LLMs in a highly sensitive negotiation context. While it was noted that completely replacing humans is almost impossible, the contribution to the analysis was indeed significant. These models allow for a faster comparison between past scenarios and current ones and facilitate the analysis of the current stances of all parties involved. These factors can provide useful guidelines for experts, allowing to locate the core issues and focus on them.

1.3.3 Trading agents

On a more economic subject, this experiment[31] investigates the capabilities of AI-agents when acting as buyers and sellers inside a market. The setup involved a series of negotiations on the prices of real-world products of different values, ranging from electronics to houses, comparing how different families of models performed on each task. The experiment followed four different agents across multiple rounds of negotiation.

- *Buyer*, the agent who wants to acquire the product at the lowest possible price. It is equipped with a maximum spending budget.
- *Seller*, the agent who is selling the product, aiming at the highest possible price.
- *Judge*, a watchdog agent which is tasked with understanding when the negotiation is over, either with acceptance or rejection.

- *Analyst*, an agent tasked with following the conversation and reporting on price changes.

The experiments were carried out both by pairing different models against each other and the results showed that negotiation capability scales with the model's complexity. It is reported that there is a strong correlation between performance on this task and other generic benchmarks for the same models. Additionally, concerning the overall performances, it was observed that trading agent often violated their constraints, with buyers exceeding their budgets and sellers going below their minimum selling threshold. Other issues involved endless negotiation loops and immediate trades without any form of discussion. The conclusions of this study suggest caution towards the application of these types of agents in any form of trade as they do not act reliably. On top of that, when they do, the result of the negotiation is heavily dependent on the model's own performance. The risk is then the creation of imbalanced trade scenarios in which whoever can afford a better model will manage to obtain the better deal. Lastly, it was noted that the poor reasoning capabilities of these types of agents when it comes to numerical problems still poses a big obstacle for their application in any economic-centred scenario.

Chapter 2

Fundamentals of Game Theory

2.1 Game theory

As presented in the previous chapter, artificial intelligence systems can become a powerful tool for improving informed decision-making, by providing additional layers of strategic reasoning. However, as large language models become prominently used, it is necessary to assess their reasoning capabilities, especially in relation to their rule-based relatives. The debate on symbolic and sub-symbolic reasoning is still open in the context of artificial intelligence research and the quest for the development of a true neuro-symbolic model is ongoing. In this context, a field of mathematics such as game theory could provide a valuable benchmark. The core of game theory is the study of strategic interaction between two or more agents which are involved in a well-defined scenario, called "game". Each decision can lead to a specific reward or punishment, usually represented by score gains and losses. The variety of games that can be defined under this premise is extremely vast and allows for the application of game theory to a wide range of fields, from the analysis of social behaviour, to economics to computer science. The following sections will be dedicated to a brief introduction to the main concepts behind game theory. Starting from a mathematical standpoint, we can define an n-player game using the following components:

- n agents (players), each indexed by $i \in N = \{1, ..., n\}$.
- Action set for each agent $A_i = \{a_i^1, ..., a_i^k\}$ where k is the number of available actions per player.
- Payoff matrix $U_i(\mathbf{a})$ for all strategy profiles $\mathbf{a} = (a_1, ..., a_n)$ for a_i in A_i.

From this moment onwards, for the sake of simplicity and relevancy to the subject, we will only consider two-player games. Definitions can easily be extended to multi-player games. The payoff matrix can be considered the core of the entire decision-making process, as it contains all the necessary information to understand the game. In a two-player game we may have two player action sets: one for p_a: $A_a = \{a_a^1, ..., a_a^s\}$ with s actions and one for p_b: $A_a = \{a_b^1, ..., a_b^t\}$ with t actions. This setup would make U an $s \times t$ matrix, where entry U_{ij} represents a strategy profile (a_a^i, a_b^j), thus containing the appropriate payoff vector $(u_a(a_a^i, a_b^j), u_b(a_a^i, a_b^j))$. To put it simply, rows represent actions of p_a, columns actions of p_b and entries the ordered payoffs received by them in correspondence to a combination of their actions.

2.1.1 Types of games

Under the previous definition, the concept of "game" can be vague, to add further context we introduce a few additional definitions. Based on the level of knowledge a player has about the game we can define:

Complete-Information Games. In these games all players have full knowledge of the structure of the game, having access to every action available to each player and knowing the entire payoff matrix. In this type of games, each agent can focus on computing the optimal strategy, which can be deterministically deduced and executed.

Incomplete-Information Games. Whenever one or more of the players does not have full knowledge of one element of the game, we have an incomplete-information game. Under these circumstances, a player may develop a series of beliefs on the missing information, i.e. a set of assumptions under which it will choose to act. These types of games analyse the strategic process under uncertainty, deploying probabilistic reasoning to assess their odds.

Based on the types of interaction the players can have and their timing, we can define:

Simultaneous Games. In simultaneous games, players make their decision simultaneously. This impedes every player from factoring their opponent's current move into their strategy as each agent acts independently of one another.

Sequential Games. In sequential games players make decisions one after the other, following a specific order. The evolution of status of these games can be easily represented as a tree, as each player is supposed to respond to the other's action.

Below, in Figure 2.1, we have a table with classical examples to illustrate this classification.

	Complete-Information	Incomplete-Information
Sequential	Chess	Poker
Simultaneous	Rock-Paper-Scissor	Betting on dice

Figure 2.1: Classification of classical games

2.1.2 Game statuses

During the evolution of a game, not every configuration and decision is equivalent. Given that game theory is a mathematical model, we can identify some noteworthy configurations which hold a special meaning in the context of the game itself. They will be introduced in the following definitions.

Nash Equilibrium. A Nash equilibrium represents a strategy profile where no player can unilaterally improve their payoff by deviating from their current strategy, assuming all the other players do not deviate from their own. In formal terms, given a strategy profile $(a_1^*, ..., a_n^*)$, it is a Nash equilibrium if, for every player i: $U_i(a_i^*, a_{-1}^*) \geq U_i(a_i, a_{-1}^*)$ for all $a_i \in A_i$, where a_{-1}^* represents the equilibrium strategy of every other player aside from i. The Nash equilibrium is an important turning point for every type of game, since it represents a situation in which coordination between players becomes necessary if they want to increase the payoff. Assuming there is a better payoff, which is true in most cases, going beyond the Nash equilibrium represents a moment of awareness for players, as they forego a self-centred strategy for a more complex one.

Pareto Optimality. A strategy profile **a** $= (a_1, ..., a_n)$ is considered Pareto optimal if there is no feasible action **a'** $= (a_1', ..., a_n')$ such that $U_i($**a'**$) \geq U_i($**a**$)$. This must hold for all agents $i \in \{1, ..., n\}$ and $U_j($**a'**$) > U_j($**a**$)$ for at least one agent j. Under a Pareto optimal status, no player can improve their current payoff without diminishing another player's. This is another significant status which, despite not being an equilibrium point, represents the best possible scenario all players can achieve by coordinating.

While a Nash equilibrium represents the best situation a player can achieve by considering only itself, the Pareto optimal profile highlights the best global situation all players can achieve together. The two conditions are not mutually exclusive, depending on games, usually reaching a Pareto optimal Nash equilibrium represents the best possible scenario, in which individual and group objectives align.

	Cooperate	Defect
Cooperate	(R, R)	(T, S)
Defect	(S, T)	(P, P)

Table 2.1: Payoff matrix for the Prisoner's Dilemma

2.2 The Prisoner's Dilemma

In this section we will introduce a staple of game theoretical analyses: the Prisoner's Dilemma.

2.2.1 The Game Premise

The Prisoner's Dilemma (PD) is an example of an information-complete simultaneous game, which can be described by its simple premise.

"Two robbers are caught by the police after robbing a bank. At the moment of the arrest, since there is not enough evidence, both prisoners would face one year in jail. Knowing this, the agents come up with a plan. While they are being interrogated separately, each suspect if offered a deal: if they confess while the other does not, they will be free, while the other will face three years in prison. There is, however, a catch: if both confess, they will face two years in prison each."

From this simple scenario, we can reformulate the problem under the previously defined terms. PD is a 2-player game. Players are named a_1 and a_2. Each player has an identical set of two actions $A_1 = A_2 = (\textit{cooperate}, \textit{defect})$. Where *cooperate* means staying loyal to the partner, while *defect* means confessing. Based on the description, a parametric version of the payoff matrix can be designed as in Table 2.1. In this form we can identify four parameters:

- R, the Reward score, representing the years of jail they would get by mutually cooperating.
- P, the Punishment score, representing the years of jail they would get by mutually defecting.

- T, the Temptation score, representing the years of jail a player would get by defecting while the other cooperates.
- S, the Sucker score, representing the years of jail a player would get by cooperating while the other defects.

According to the classical formulation, the defined score would need to follow a specific ordering: $S > P > R > T$. However, following game theoretical convention, increasingly better outcomes are associated to the higher score. This decision stems from the necessity of uniforming each game as a reward maximization problem. According to this, the scores are reordered as $T > R > P > S$.

2.2.2 One-Shot PD Games

To analyse the Prisoner's Dilemma properly, it is important to focus initially on its one-shot format. In this case two players play one single round. Under this assumption, it is easy to spot that defection is always the best strategy. This can be easily shown since:

- Supposing a_2 defects, a_1 is better off defecting, thus getting P instead of S, as $P > S$.
- Supposing a_2 cooperates, a_1 is better off defecting, thus getting T instead of R, as $T > R$.

Considering this, by the previous definition, we can conclude that, in the case of a one-shot game, the natural Nash equilibrium is *(defect, defect)*. However, given that *(cooperate, cooperate)*, would lead to the best possible outcome globally, the equilibrium strategy profile is not Pareto optimal. Starting from this premise, since they will be useful in the later stages of this dissertation, we will introduce two other parameter orderings. By changing the orderings, the winning strategies change, giving rise to two adjacent games, with different outcomes. Both of the following versions will make little to no sense in the context of the original game and represent, more than anything, theoretical models. They do, however, provide interesting investigation cases.

Prisoner's Delight. In this scenario, we have $S > P > R > T$. The payoffs are inverted from the usual setting.

- Supposing a_2 defects, a_1 is better off cooperating, thus getting S instead of P, as $S > P$.
- Supposing a_2 cooperates, a_1 is better off cooperating, thus getting R instead of T, as $R > T$.

With these new premises, we have shifted the Nash equilibrium to *(cooperate, cooperate)*, given the complete shift in payoffs, the Pareto optimal strategy profile becomes *(defect, defect)*. This scenario presents a strong incentive towards cooperation.

Prisoner's Confusion. In this scenario we impose $R > T > P > S$, the noticeable change being that now $R > T$. Again we can analyse the scenarios.

- Supposing a_2 defects, a_1 is better off defecting, thus getting P instead of S, as $P > S$.
- Supposing a_2 cooperates, a_1 is better off cooperating, thus getting R instead of T, as $R > T$.

Under this new assumption we have two Nash equilibria, since both *(cooperate, cooperate)* and *(defect, defect)* represent valid individual strategies. The main difference between them is that, between the two, *(cooperate, cooperate)* is also a Pareto optimal.

2.2.3 Iterated PD Games

The analysis perspectives of a one-shot Prisoner's Dilemma game are, however, quite limited. To properly explore the implications of this experiment, we must introduce its iterated variant. The premise for the Iterated Prisoner's Dilemma is simple: the two players will play for a number N of rounds, having a certain degree of memory of the past interactions. Depending on the experiment, the window of memory may be limited or indefinite. By reasoning upon previous interaction, a player agent can react to previous decisions of its opponent, thus developing a strategy for decision-making. One of the important mathematical premises for the classical IPD is the $2R > T + S$. This condition is necessary to allow cooperation to be the better choice in the long run. As discussed before, if no further interaction is present, defection is the optimal choice. When taking into account multiple rounds, however, agents have the possibility to reflect on past decisions and coordinate together to achieve the Pareto optimal profile. The rise of a mutually cooperative behaviour does not change the fact that the Nash equilibrium still remains set on mutual defection. The difference is that repeated interaction may lead to the creation of a "tacit agreement" between players, to ensure a better score for both. From this perspective we can foresee the social interpretation this experiment has. Humans live in a society where repeated interactions with the same individuals are frequent: for the benefit of everyone, cooperation becomes part of the agreement that holds society together.

One of the most important studies on IPD strategies is *The Evolution of Cooperation* [4] by R. Axelrod. During the course of this famous study, a tournament was held between multiple rule-based agents, each one equipped with a strategy. All agents would compete in a setting governed by natural selection, where players with lower scores would end up discarded and those with higher would go on to replicate. The strategies adopted were designed by Axelrod's colleagues and the tournament became a significant milestone in the study of how cooperative behaviour can emerge from natural selection.

Starting from this study we can identify some of the most noteworthy characteristic a strategy can exhibit:

- Nice, a strategy which does not involve defecting before the opponent does.

- Retaliating, a strategy which answers to a defection with a defection.
- Forgiving, a strategy which returns to cooperation when is met again with cooperation.
- Non-envious, a strategy which does not aim at achieving a higher score than the other player.

Upon these characteristics we can classify all strategies, depending on whether they exhibit this trait. Below, we will present some of the most common ones and their traits.

AC — Always Cooperate. This strategy is very simple and involves always choosing the *cooperate* option. It is nice, non-retaliating, forgiving and non-envious. Given its simplicity, it was deemed as highly exploitable, since it does not react in any way to repeated defections. Its most favourable matchup is another AC, since they would never take advantage of each other.

AD — Always Defect. Dual strategy of the previous one, it involves always deciding the *defect* option. It is a non-nice, retaliating, non-forgiving and envious strategy. Although it may seem more solid, given that it focuses on the Nash equilibrium strategy. Nonetheless, it was revealed as inefficient, since it leads to very modest results when faced against other AD or any retaliating strategy.

GT — Grim Trigger. A slightly more complex strategy with respect to the previous, GT, involves always opting for *cooperate* until it is met with defection; from that point onwards, it would only choose *defect*. This is an example of a nice, retaliating, unforgiving and non-envious strategy, with its core flaw being its lack of forgiveness. The spiral of mutual defection this strategy follows leads to lower scores when facing occasionally defecting strategies, since it does not give them a second chance.

TFT — Tit for Tat. This strategy was the ultimate winner of the Axelrod tournament, and it is quite simple, it starts with *cooperate* and proceeds by mirroring the previous action of the opponent. By acting this way, TFT is nice, forgiving, retaliating and non-envious. In fact, all the characteristics were defined starting from the behaviour of this one. TFT is particularly effective because it is capable of kickstarting mutual cooperation with every willing strategy, punishing any attempt of exploitation but acknowledging a return to a cooperative behaviour.

It was noted that the initial population plays an important role in the results of a tournament, even a TFT can be overwhelmed in case the vast majority of the players use an exploitative strategy.

2.2.4 Real-World Applications

The Prisoner's Dilemma is a famous experiment way beyond the field of mathematics for its versatility of application to multiple real-life scenarios. In the field of psychology, parallels have been drawn between long term relationships and games such as PD. In the construction of trust mechanisms between two individuals, a certain degree of mutual cooperation is needed for the relationship to prosper. Similarly to a TFT strategy, forgiveness is a significant factor, as well as retaliation, which is necessary to set boundaries and avoid emotional abuse. In economics, where game theory is integrated as a tool for strategic-decision making, many scenarios mirror the one observed in the PD. Specifically, a comparison can be noticed in the context of two competing companies. When setting prices for their products, each company may be tempted to lower prices in order to win over customers, this would trigger a response from its rival, which may lower the prices even more. Following this trend, both companies would end up earning increasingly less money, thus damaging each other in the long term. Instead of fighting, the alternative option could be to agree on a common price, which would benefit both, although at the expenses of the customers. Lastly, even the field of international politics is not exempt from parallels with the Prisoner's Dilemma. In a scenario where all countries want to ensure their security, the may be temped to develop stronger weapons. As observed in the past, this would lead to an armament race which may end up making the entire world less safe overall, including them. Even in this case, a cooperation agreement, such as the *Treaty on the Non-Proliferation of Nuclear Weapons* could avoid a worse scenario.

2.3 Behavioural Game Theory

One of the core assumptions that concerns game theory is about the rationality of the agents. When analysing the best strategy profile in any game scenario, it is always assumed that the acting player will make a choice guided by a principle of efficiency. In the context of the Iterated Prisoner's Dilemma there are compelling arguments for both choices, but they can always be explained by a desire to obtain the best possible score. When analysing human behaviour, however, rationality may not be the guiding factor for decision-making. It is well documented that humans can act irrationally even when faced by scenarios where a logical reasoning could suggest the mathematically optimal outcome. Behavioural game theory[6] aims at investigating the psychological factors that affect human behaviour in contrast to rationality. To do so it uses scenarios drawn directly by game theory. There are a multitude of reasons which can influence a human player even in a controlled environment such as a game theoretic experiment. Using the IPD as an example: a player may choose to cooperate not by deducing that it is the optimal choice in the long run, but simply because it considers morally wrong to betray the trust of a hypothetical partner. On the other hand they may select to defect, arguing that cooperating with police authorities is a more socially acceptable behaviour. An interesting consequence of the irrationality of human behaviour is that its effects were propagated in our actions, and are thus registered in the data we produce. As a consequence, when machine learning models are created and fed real-world data, they may learn to act more irrationally as part of the inherent bias. This phenomenon is well documented in LLMs, since they are trained on textual data, and it is shown in their innate inability to play highly strategic games, such as chess. As research progresses on the reasoning process of these systems, it seems apparent that, unlike classical AI, their behaviour shows clear signs of irrationality [11]. Given these premises, behavioural game theory may be a better tool to analyse the behaviour of LLMs since, contrary to common assumptions regarding AI, they are prone on choosing based on their "instinct" rather than through rational deliberation.

Chapter 3

LLMs and MAS: state-of-the-art

3.1 Transformers and Large Language Models

Among the recent developments in the field of Artificial Intelligence studies, transformers and LLMs are certainly the ones that contributed to its attention surge. Introduced in 2017 through the already foundational paper *Attention is all you need*[27] transformers revolutionized the field of natural language processing. This publication introduced the usage of the *attention* mechanism inside a neural network architecture, which proved extremely effective at handling the task of probabilistic sequence generation. The proposed architecture, named *Transformer*, is currently the foundation structure for all Large Language Models. This section will delve into the current available technologies and provide some insight on the current landscape of open weights models, analysing briefly their design choices.

3.1.1 Attention Mechanisms

Attention is the core mechanism for the correct functioning of any transformer architecture. The basic idea behind it is to allow the model to focus its "attention" only on certain parts of the input sequence. To do so, the internal structure is designed to account for soft weights, which are used only in the forward pass of the generation and are strictly associated to one token of the sequence, before changing for the next one. With the attention mechanism, each token is factored inside the context of the overall sentence and affects the probability distribution of the next token. This process continues until the end of sentence token is predicted as the most probable. The core of the attention mechanism resides in the comparison, through normalized dot product, of the ordered query matrix Q (containing the input token), with the K matrix which contains an unordered list of all the possible embeddings. V contains the actual values, weighted by their attention scores.

$$Attention(Q, K, V) = softmax(\frac{Q \cdot K^T}{\sqrt{d_k}}) \cdot V$$

Starting from this general premise, different refinements were designed throughout the years, below, we'll list some of the most prominent.

Masked Attention. This improvement on the attention mechanism works by "masking" all future tokens inside the query sequence. By assigning 0 to every attention score the transformer can focus its attention on the previous tokens of sequence, without being impacted by the incoming ones. This practice is more efficient and establishes a more causal dependency in token generation.

$$Attention(Q, K, V) = softmax(\frac{Q \cdot K^T}{\sqrt{d_k}} + M) \cdot V$$

Where M is a strictly upper triangular matrix, with 0 on and below the diagonal and $-\infty$ above it.

Multi-Head Attention. By introducing multiple sets of attention scores, it is possible to capture multiple types of connections inside the sequence. This way the same token can be associated with multiple ones at the same time. To achieve this it is necessary to introduce multiple matrices sets, each called an *attention head*, for a total of h. The resulting value vectors will be concatenated in the end.

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h) \cdot W^O \text{ with}$$
$$head_i = Attention(Q \cdot W_i^Q, K \cdot W_i^K, V \cdot W_i^V)$$

Where all the W matrices are additional parameter matrices.

Grouped-Query Attention.[2] Given the computational cost of multi-head attention, another refinement involved the grouping of multiple attention heads under the same group of weight matrices. This reduced the number of necessary parameters, keeping the same number of heads and thus preserving the possibility of focusing on multiple part of the sequence. Ablation studies suggest that this grouping does not affect performances significantly, while reproducing costs on the other side. Specifically, in GQA, the grouping is performed on a K and V matrices. It must be noted that GQA, while introducing a useful cost-saving technique, was considered mostly a work-around. With the objective of preserving the effectiveness of MHA while limiting its costs, it became necessary to develop more structurally sound techniques.

Multi-Head Latent Attention. Keeping in mind the objective of memory usage reduction, Multi-Head Latent Attention (MLA) saves space by caching reduced forms of the K and V matrices. Specifically, instead of grouping different heads under the same key and value matrices, the idea is to compress them in to a lower dimensional space before caching. This is done both during training (where even queries are reduced) and during inference, allowing to save memory but avoiding sharing the same weights for different heads. Comparison with classic MHA architectures showed similar if not better performances, while introducing a significative reduction in computation cost.

3.1.2 Mixture of Experts

Usually, the terminal part of a neural network model is constituted by a feed-forward dense layer, and transformers are no different. A recent improvement in this area involved the introduction of the Mixture of Experts (MoE[10]) layer [22]. What MoE does is substituting the single feed-forward layer with an ensemble of multiple feed-forward layers, each specialized (i.e. expert) in a specific type of generation. This change, considering that the number of experts can reach the hundreds, would entail in massive increase in parameter count. To counter this the access to each expert is gated by a router. The router component activates only some experts for every single token that is being processed. This allows the model to learn more during the training phase, thanks to the augmented parameter space, but still allows for a faster inference, given that only certain parts will be used for computation. The training activity has to teach the model when to invoke each expert, but this structure allows for way more flexible models. MoE architectures, given their partial usage of their total parameters, are thus regarded as sparse models. Initially, architectures that used MoE had a reduced number of larger expert layers, usually invoking a low number of them (2 or 3) at the same time. Further studies changed this pattern, by using a higher number of smaller experts, allowing for the activation of more of them simultaneously. The last refinement of the MoE structure involved the introduction of a shared expert, a small expert layer which is always active and is assisted for each token by the router selecting other experts. The benefit of the shared expert is easily deducible: while commonly repeated pattern are learned by the shared expert, specialized ones are learned by each expert, allowing them to contribute only when necessary.

3.1.3 Overview of three current OS models

After the introduction of some of the currently most relevant technologies for the construction of modern transformer architectures, we will present three of the currently available open-weight models. Their choice is motivated by the availability of their architecture, their relevance in the context of the experiments of Part II, and ultimately because they exhibit the implementation of techniques seen in the previous section.

MISTRAL 3.1. One of the key achievements of this Mistral model is its inference speed. This was achieved through a combination of different factors, such as a custom embedding and reduced dimensions for the K and

V matrices. Mistral 3.1[16] utilizes the GQA technique for attention, which additionally increases its efficiency. It is regarded as a good trade-off model, given its lightweight nature with overall effective performances. As an open weight model, its lightweight nature is essential to ensure it can be run locally on smaller machines. It is worth mentioning its function calling capability, which can greatly expand its reasoning potential and allows for its usage in complex agentic architectures.

LLAMA 4 Maverick. The LLAMA 4[26] is an all around multi-modal generation model with 17 billion parameters. Considering its text generation variant, we can notice that it adopted the MoE layer to achieve a more efficient parameter usage. In doing so, it followed the earlier approach, with a small number (usually 2) of large (8192 units) experts active at the same time over a total of 128. It must be noted that not every transformer block uses MoE, opting instead to alternate MoE and dense layers between each block. Considering its adopted attention mechanism, this model uses GQA to further reduce its weight size and incentivize its adoption in small scale systems. During its training process, distillation from the huge Llama Behemoth 228B was adopted, with the objective of teaching part of its larger knowledge to a smaller model. Performances of this model classify it as comparable to many closed-weight models in several benchmarks, specifically reasoning-oriented ones.

GPT-OSS. Being one of the few open-weight models[12] released by OpenAI, it comes into two different versions: 20B and 120B. From an immediate comparison, it is possible to observe that these models focus more on width than depth, having a larger embedding size (2880) and an intermediate expert projection of 2880 units. Additionally, this model is equipped with a higher number of attention heads, which reduces the inference time, by parallelizing computations. Again, GPT-OSS utilizes the MoE technique, opting for a total of 32 experts, having at most four active at any given time. Even in this case, no shared expert is utilized. As the previous example, even GPT-OSS utilizes

GQA to reduce computation costs. Both of these variants are equipped with function calling capabilities and support agentic integration.

3.2 Agentic LLM Technologies

Large Language Models are undoubtedly a powerful tool on their own, given their ability to analyse different media and produce responses through inference. A recent development to further increase their capabilities saw their integration into the well-established field of agentic modelling. The objective of this combination is to develop intelligent assistants, capable of handling multiple complex tasks.

In computer science, an agent can be defined as: *"A computer program that performs various actions continuously and autonomously on behalf of an individual"* [32]. When defining an agent, a special focus must be given on the concept of autonomy, which stems from the fact that they can act independently of human supervision. In classic symbolic AI an agent is programmed by giving strict guidelines on how and when to act. In those cases the behaviour could be explained by analysing the scenario the agent was in: depending on the situation, conditional programming would lead to the execution of certain parts of code instead of others. With modern LLM-based agents the situation is much different, since they lack the deterministic predictability of rule-based ones. The increased benefit of this new family of agents lies in their flexibility, which comes from their generalist approach to problem-solving. And while it is true that LLMs lack natively the specificity to handle more complex tasks, tool usage may be the key to address this issue.

3.2.1 LLM Agents

LLM agents represent an attempt at unifying the persistent and autonomous nature of classical software agents with the expression capabilities of large language models. To achieve this result, LLM agents are equipped with two fundamental components: memory and function. The addition of persistent memory allows agents to save the results of past interactions and load them as context in the future. This extends the limitations normal chatbots encounter when dealing with users, as there is a limit to their context window. With the introduction of tools, agents have the possibility to rely on a piece of code that produces explainable and consistent results. By executing code, agents can greatly compensate their lacking reasoning capabilities, especially when it comes to dealing with numerical values. Since tools represent a symbolic component inside the LLM reasoning process, they greatly increase the explainability of their decision-making.

Generally speaking, an agent's workflow is composed primarily of three main parts.

1. Goal initialization and planning, consisting of the initial setup provided by the developers through training and by the user with the prompting. Starting from these elements and with knowledge of its available tools, the agent can develop a plan to achieve the required goal.

2. Reasoning with tools, given that the agent lacks individually the information to provide a full answer it must rely on tools to update its knowledge. Using tools and obtaining results from them, the agent gains additional context and becomes capable of providing proper answers, thus fulfilling its goals.

3. Learning and reflection. Once the goal is achieved, the agent collects feedback and information on the concluded action, storing it in memory. The next time the agent will face a similar task, it could utilize the previous knowledge to perform better, taking into account previous successes and mistakes.

Depending on how structured the agent's internal architecture is, it is possible to distinguish different types of LLM-based agents .

Simple Reflex Agents. This type of agent is extremely simple, as it does not have memory and can not interact with other agents to gain additional information. An agent of this type can react if a certain condition is met and will not respond properly if the request is beyond its capabilities. Lack of memory means that there is no possibility of learning from past tasks, and it is extremely fragile against unknown scenarios.

Model-based Reflex Agents. These agents are equipped with memory and use it to maintain a model of the world they are interacting with. Once new information is provided, the agent updates its internal memory and uses the new knowledge to react to a change in conditions. The acting capabilities of these agents is still heavily limited by what they can do, since they can not communicate with external systems.

Goal-based Agents. Similarly to Model-based agents, they have an internal memory state but upgrade it by having a set of goals they are set to achieve. The conjunction of these two separate contexts allows them to make plans based on their current model of the world with the objective of fulfilling their goal. The possibility of planning ahead based on knowledge is the deciding factor for a more structured approach to decision-making.

Utility-based Agents. These agents are similar to the previous Goal-based agents, but present a significant difference with the introduction of a utility function. The utility function is an additional tool that help the agent determining how “good” an action is for reaching its goal. Since the utility function can be structured to factor many aspects and the agent is programmed from the start to maximize the overall utility, it can use it as a guide to evaluate the actions it takes. This self-evaluation capability, combined with the memory, allows the agent to create a cumulative experience and improve significantly its reasoning over time.

Learning Agents. The complex internal structure of these agents allows it to achieve a more consistent form of learning than previous ones. The architectures exhibit four main components:

- Learning, a memory module to update its knowledge base with previous interactions.
- Critic, a feedback provider which judges the utility of actions and responses
- Performance, the action selector tasked with choosing the right decision based on available knowledge
- Problem generator, the action proposer which presents all possible actions available in a situation

3.2.2 Prompting techniques

One of the core parts of the initialization process of a LLM-based agent is prompting. As discussed before, each agent has a set of temporary weights which are affected by context and influence the generation process. This is called In-Context Learning. Prompt-engineering involves the deployment of different prompting techniques with the objective of finding the most adequate for the execution of a task. This applies to chatbot prompting as well as agent modelling. In the case of LLM-agents, the initial prompt is particularly important, since it defines the agent's goal and its starting set of knowledge. A well documented phenomenon concerning LLM behaviour is their fragility against prompt modifications. The same task, phrased differently, can be interpreted and executed by the same agent in a completely different way. Considering this situation, it is not surprising that an ample range of prompting techniques was developed in the last years, each oriented to a specific task. Below, we will report a few of the most recent and relevant to the topics at hand. As a basis for the definition of these techniques, we will refer to a recently published survey [21] which documents a taxonomy of 58 different prompting techniques. Among these, the focus will be specifically on text generation models.

Zero-shot Prompting. This is one of the most basic, yet effective, prompting techniques available. The basic idea is providing the agent with a description of the task it must work on. The description must be the most precise possible to help the agent figure out what the goals are. Having no examples, it is up to the agent's capabilities to understand what it is being asked to do. One of the most common usages of this type of prompting for agent modelling is *Role Prompting*, where the input describes the agent which kind of role it should have. In this case the objective is to condition the agent's future responses, by forcing it to act and behave claiming to be someone. Another usage is *Emotion prompting*, where a question is presented incorporating psychologically relevant phrases, to elicit a more elaborate and careful response. *Self-Ask* is another of these techniques, based on the idea of asking the agent if it requires additional questions before answering. This is done to first elicit an answer to the clarification questions and the initial one.

Few Shot Prompting. In this style of prompting the question is paired with one or more examples of appropriate answers. The goal is to guide the agent by telling what it needs to do, possibly providing various examples to avoid overfitting the problem. It must be noted that the way examples are presented affects the model performances, weighing more the first examples given and favouring certain formats. The differences between forms of Few-Shot prompting resides in how the examples are chosen. Among these there is *K-Neighbour clustering*, to cluster examples and provide one example for each cluster. Another refinement, called *Prompt Mining* works by analysing how questions and answers are posed inside the corpus, finding out the most common structures and using those.

Chain-of-Thought Prompting. This family of techniques aims at enhancing the model's reasoning capabilities by pushing it to reason more during the generation. CoT can be achieved by leveraging zero-shot techniques, like adding phrases to incentivize reasoning, or by suggesting an initial reflection on base concept, such as in *Step-Back prompting*. There are also some few-shot techniques, such as *Contrastive CoT prompting*, where multiple examples of reasoning are provided, inserting both accurate and inaccurate ones.

Decomposition Prompting. Similarly to CoT, decomposition prompting aims to help the agent reason by proposing ways to divide the problem into smaller ones. *Least-to-Most prompting* guides the model to decompose the task in simpler ones, without solving them. Then, it allows for a sequential solution based on the results of each one, until the goal is achieved. Another example is *Tree of Thought (ToT)*, which incentivizes the creation of a tree of potential steps and the exploration of said tree, with each branch choice leading one step closer to the solution. The last example is *Metacognitive Prompting*, which tries to simulate human cognition by breaking up the resolution into five steps: question clarification, preliminary judgement, evaluation of response, decision confirmation and confidence assessment.

Ensemble Prompting. These techniques involve the usage of multiple prompts to generate more than one solution, aggregating all results into one, whether by summarization or by a majority vote. An example of these techniques is *Self-Consistency*, which prompts the LLM multiple times with a non-zero temperature to elicit diverging responses and outlining the correct solution by using majority voting. Built on this is *DiVeRSe*, where multiple prompts are generated for the same question, using them to perform Self-Consistency to generate multiple reasoning paths. Each path is the scored for each step to determine the final answer.

Self Criticism. This last set of techniques involves the usage of the agent itself to generate an evaluation of its own output, which is then used to further refine the answer. Among these we have *Self-Calibration*, which prompts the LLM to answer, then asks it if the provided answer (which is attached in the question) is correct. The objective is to obtain a confidence level for the answer. In *Self-Verification*, the model is asked to generate an answer then, after masking the parts of the original question, it is asked to reconstruct it based on the available parts and the answer. Lastly, we can mention *Cumulative Reasoning*, where the model is prompted to generate several steps of the reasoning and is then asked if they are enough to provide a solution or not. Based on the answer, the model is prompted again until it produces the entire answer.

3.2.3 Tool usage

The idea of enabling agents to use tools to enhance their reasoning capabilities was repeatedly mentioned throughout this section; it is therefore necessary to address this point since it will be relevant in the following chapters. It must be noted that not all LLMs are capable of natively using tools and this capability is usually specified in their model cards. To train a model to be able to utilize tools, a specific fine-tuning with dedicated datasets is needed. Tools are an essential part of building an agentic system, they are necessary to complement its reasoning capabilities and to provide real-time data when required, since the agent has native knowledge only of its training data. Now, LLMs are designed to produce some form of textual output and are thus incapable of running code by themselves. This is where the agentic nature of the architecture becomes essential, to provide the LLM with the necessary framework to answer its calls. If an LLM was trained to utilize tools, based on its context it may recognize that its current knowledge is insufficient for the task and decide to rely on a tool. To call a tool, the LLM must express it in a specific syntax, which will be recognized by the underlying agentic structure, which will invoke the function and return its result for further processing.

A tool is a software component of the agent which is external to the LLM but which can be "described" to it through its context. The description of the tool must be provided using text, since it is the main reasoning medium for the LLM. Specifically, the description must specify two key aspects: tool purpose and expected parameters. The textual description of the tool is essential for the LLM, which is completely unaware of the underlying implementation and simply expects to obtain some type of result when it specifies the signature of a function with the requested parameters. The description of the tool will be injected directly in the system prompt of the agent, in addition to any initialization parameter. This way the agent will be always aware of the tools at its disposal before answering. In the context of tool usage a notable mention is the Model Context Protocol, an open protocol which is becoming a new standard for tool invocation and that is already being supported by many models. The main contribution of MCP is providing a common interface for numerous APIs, which allows for the development of modular and more manageable architectures. Lastly, it is important to dedicate a few words to the different prompting techniques that can be deployed to leverage a model's tool usage capabilities. The first one to mention is *Modular Reasoning, Knowledge and Language (MRKL)* which utilizes a router to invoke multiple tools and obtain different responses to be combined into the final one. This was mentioned in one of the founding papers on tool usage [20]. The second is *Self-Correcting with Tool-Interactive Critiquing (CRITIC)*, where the LLM is prompted first to respond without using tools, then to critique its answer and finally to correct it by allowing tool usage.

3.3 Frameworks for MAS and LLMs

As seen in previous sections, Large Language Models can become the centrepieces of evolving and powerful agentic programs. While equipping them with tools and resource access is a good way to improve their overall performance, there are still limitations at play when it comes to their ability to display even more complex behaviour. One of the immediate responses to this issue was to join multiple agents into larger, multi-agent systems. Arguing that multiple heads work better than one, this approach seems reasonable. However, making sure these systems interact organically is a task in of itself: after all most LLMs are designed for the interaction with a human user, not others of their kind.

The interaction between multiple LLM agents inside the same system can be structured in many ways, following different architecture patterns. Usually, a system of this type is assigned a task, but the way agents tackle it can vary significantly. For example, the problem may be decomposed by one agent, while others are individually dispatched to solve each single part and report back to the orchestrator. In other cases, multiple agents may be deployed together to find a solution, each with a different set competences and tools. In this last case, agent coordination is essential, since they share some resources and environment. Similarly to a classical *Blackboard* pattern, each agent must recognize when to make a contribution and what to do. A subtype of this collaborative approach to LLM interaction which leverage their inherent communication capabilities is multi-agent debate. Given its relevance to the experiments of this thesis, a brief introduction to the concept will be presented, as well as a few technologies that support its implementation.

3.3.1 Agent Debates

Multi-Agent Debate (MAD) is a technique which originated from the desire to improve the overall truthfulness and accuracy of information produced by LLM systems. Originally, it was designed in the context of question answering, but it recently found applications in many other use cases. The basis for this approach in the field of Q&A is straight-forward: a question is introduced to a system of agents which share common access to a memory structure representing their *conversation*. The question prompt represents the starting point of it and each agent's contribution is logged onto it, thus increasing contextual information for all agents, which will factor previous messages in their deliberation. Every agent inside the system is initialized through a prompt visible only to it, this is usually referred to as *persona*. The persona is an internal factor which, together with the tools an agent may be equipped with, will condition the way it responds. Another aspect which heavily conditions the way the debate evolves is the speaking schedule, which governs how agents are supposed to interact, either by letting everyone speak whenever they prefer or by fixing turns. The last factor to consider is the conversation termination condition: this is closely tied to the objectives of the entire system. In some cases agents debate freely for a specified number of rounds. In others they need come to a conclusion through some form of consensus mechanism. Whichever the interaction conditions are, a key parameter in the construction for a system of debating agent is the prompting. LLM agents can be frail and, if the scenario is not described correctly, the conversation can take unexpected turns. Based on this paper[23] on the subject, we will present a few standard prompting techniques appropriate for this setting.

Society of Minds. A simple setup where similar agents can speak and add their opinion to the conversation, with consecutive answers taking into account the newly extended context. A variant of this approach sees each agent speak based on a summarization of all previous interaction instead of the full text.

Multi-Persona. This strategy is designed to encourage diverging outcomes by initializing each agent with a different persona. In the basic case there were two: an affirmative agent (angel) and a negative one (devil). The overall exchange between agents is regulated by a judge, which oversees the debate, understanding when a solution has been produced or ending early if satisfied with the debate.

ChatEval. This strategy actually presents three different cases. In *one-on-one* agents are prompted sequentially on the same question, with each having the previous' answers available in the prompt. In *simultaneous-talk* all agents concurrently generate responses ignoring agent order. Lastly, *simultaneous-talk-with-summarizer* is similar to the previous one, but the total conversation is summarized before every new addition.

3.3.2 Technologies for LLM MAS

Making agents interact with each other is an interesting strategy to leverage their potential, but it represents a problem from a software perspective. While LLM agents are normally handled through APIs to access them locally or from remote, their integration into larger systems requires a few computational challenges. Unsurprisingly, in the last years we saw the emergence of several implementation frameworks that supports the development of these systems with various focuses and approaches. The objectives of these software components is to provide codebases and reliable APIs for developers, in addition to managing the interaction between the agents and their environment. This section will give a short overview of some of the most renowned and currently used frameworks for the definition of LLM MAS.

LangChain & LangGraph. LangChain[14] is an established open-source framework for the development of LLM agents through a simple yet effective interface which abstracts from the underlying model API. The architecture allows for the definition of tools as functional elements of code that can be equipped easily to agents during the initialization. The system is designed to account for memory and supports numerous instruments to manage the agent's lifecycle. Together with a dedicated infrastructure, called LangGraph, LangChain supports the creation of larger workflows of agents which can be coordinated through an orchestration graph. This decision allows a great degree of freedom in designing the topology of the system, since communication channels between agents are managed individually as edges. This degree of customizability allows for a careful management of the resources, which is essential in the development of large scale applications. While requiring some low-level programming skills, the overall framework is designed to support professional and stable applications, with support for reliable testing and debugging.

CrewAI. On another end of the spectrum, CrewAI[7] is an open-source platform that supports interaction of agents through a mostly no-code interface. Specifically, this framework is centred around two concepts: *Flow* and *Crew*. A flow is the orchestration abstraction which manages the entire activity of the system. It governs the execution of tasks through loops and branches, invoking crews when required. It represents an element of conditional programming since it is based on trigger conditions and input responses. A crew represents the computation unit and is a team of agents that can be assigned a task. Inside a crew each agent can be initialized with goals and tools. Inside a crew agents are designed to work based on their capabilities, with coordination attributed to the single crew unit. This framework represents an easy-to-use tool designed for a wider audience, since it allows the creation and management of entire systems using mostly natural language. Given its high-level nature, there are less customizability options in terms of software. It can be a powerful tool for developing large applications without having to worry about implementation details.

Autogen & Ag2. The last framework presented is the one which ended up being adopted to realize the experiments of Part II. Following a split between the original authors and Microsoft, there are currently two diverging branches of this architecture, Autogen[3] and Ag2[1], with the latter being the chosen one as it supports backward compatibility. Aside from some name discrepancies across the two, both alternatives are still open-source, and given their common origin, share most of the theoretical bases. The main focus of Autogen is the conversation between groups of agents, and it provides a suite of tools to allow the definition of this specific kinds of workflows. Namely, there are two main essential components inside an Autogen program: *ConversableAgents* and *Chats*. ConversableAgents are Autogen's agent abstraction, they are designed to be model-agnostic, being initialized by an `LLM Configuration` and a system prompt. Given the focus on communication, they present methods to send and receive messages, to interact with the conversation and process responses individually. On top of that each agent can be equipped with tools to enhance its reasoning capabilities. Chats are the conversation abstraction and represent the channels through which agents can interact with each other. They are highly customizable, allowing for predefined scheduling strategies or user-defined ones. Messages can be intercepted and processed both synchronously and asynchronously, allowing for a complete control of the communication flow. Given its specialization in agent conversation, as well as the flexibility of the system overall, Autogen makes for a reliable tool for building experiments centred around these topics, with a nice trade-off between scalability and expressivity.

Chapter 4

Game Theory and LLMs: Related work

4.1 Playing games with LLMs

This last chapter of Part I will be dedicated to the brief presentation of a series background studies to introduce the experiments of Part II. As discussed previously, the usage of game theoretic models as benchmarks for testing the decision-making capabilities of LLMs is well established. Consequentially, an appropriate amount of related literature was published. Game theoretic experiments represent an important group of scenarios because they allow to investigate and assess the rationality of whoever is playing them. An agent can be defined rational if it chooses the action which maximizes the chance of achieving its goal. In a game theoretic experiment, where gains and losses of points represent a direct measure of the quality of decision, the optimal choice can be objectively computed. Therefore, it becomes easier to spot irrationality as any form of deviation which leads to the maximization of rewards. Now, Large Language Models are not necessarily rational actors, especially compared to any kind of rule based agent. This is because their behaviour is the results of a statistical inference with the objective of generating the most realistic human interaction possible. To achieve this, they had to be trained on human conversation and interactions, which leads to reflecting the biases and reasoning fallacies of humans. In this context, to properly investigate this phenomenon, it is necessary to develop the appropriate instruments, such as benchmarking tools. Below we will present a well-developed benchmarking tool for assessing LLM behaviour in game theoretic scenarios. The following sections will address other relevant experiments with some classic games and the prisoner's dilemma specifically.

4.1.1 GTBench

This 2024 paper[8] introduces a benchmarking tool called GTBench. As stated by the developers, the objective is to provide a structured instrument to test the optimal decision-making capabilities of different LLMs in a wide range of games. The paper first introduces an additional taxonomy for games and proceeds to classify each of the ten analysed games according to it. Specifically, games are classified mainly based on three factors:

- Information Completeness vs Incompleteness
- Dynamic vs Static scenario
- Probabilistic vs Deterministic gameplay

The games chosen for the benchmark are: *Tic-tac-toe*, *Connect-4*, *Breakthrough*, *Liar's Dice*, *Nim*, *Kuhn Poker*, *Pig*, *Blind auction*, *Negotiation* and *Iterated prisoner's dilemma*. For each game, to properly evaluate the reasoning capabilities of the models, two match-ups were designed: *LLM vs LLM* and *LLM vs Conventional solver*. In addition to a basic prompt, all LLMs are initialized using using multiple prompting techniques, such as *CoT*, *Self-Consistent CoT* and *ToT*. For conventional solvers, a basic Monte Carlo Tree Search was deployed. To evaluate the performance of each model, two main metrics are used: normalized relative advantage (*NRA*) and classic ELO ratings. After the presentation of the overall evaluation framework the benchmarking system was tested comparing a wide range of models, accounting for commercial and open-source ones. Analysing the results of LLMs vs Conventional solvers it found that LLM struggle incredibly when it comes to deterministic games in some cases never managing to win, such as in Tic-tac-toe. This was shown to be independent of the underlying model. On the other hand, the results on probabilistic games were more balanced, with both types being comparable. This seems to suggest that LLMs seem to handle stochastic inference better than logical deduction. Considering LLM vs LLM, it was observed that models pre-trained on code generation performed generally better than others. Additionally, considering different methods of prompting, an interesting pattern was observed. While more complex models seemed to improve their capabilities when prompted with CoT or ToT, simpler ones saw their performance degrade when faced with more elaborate prompting strategies. Another comparison that was drawn is between commercial against open-source models, with the first usually performing better. Overall, this paper can be considered a good starting point for verifying the reasoning capabilities of LLMs through game theoretic experiments.

4.2 GT Experiments with LLMs

In the following studies, LLMs are subjected to some classical game theoretic scenarios, which help to underline some of the most characteristic traits of their reasoning capabilities and some of their most glaring flaws.

4.2.1 Beauty Contest Games

In this publication[15] about Keynesian beauty contest games, the authors investigate the reasoning capabilities of LLMs. This analysis is aided by the usage of Nagel's level-k classification for thought process. According to this hierarchy, which is easy to compute when it comes to this specific game, a player's cognitive capabilities can be graded by levels.

- Level 0, actions are chosen without considering all other players.
- Level 1, this player decides assuming all other players are level 0.
- Level k, the player assumes all other players are at level k-1.

According to this classification, in the context of this game, humans are classified between level 2 and 3. One-shot and iterated versions of this game seem to collocate LLMs between level 0 and 1, suggesting very limited awareness of the context. It was shown, however, that in repeated games LLM agents progressively got better, becoming more strategic, but never surpassing level 2. It was also noted that, when grouped with fixed strategy agents, LLMs managed to learn more quickly and to find the Nash equilibrium faster.

4.2.2 GT to uncover LLMs' biases

This[11] paper investigates the potential biases LLMs can manifest when it comes to approaching game theoretic experiments. Specifically, it identifies three main ones.

1. Positional Bias, as they seem to prefer actions depending on how they are positioned in the prompt.
2. Payoff Bias, as changing the labels of actions to reflect their effects on point gains affects the choice.
3. Behavioural Bias, as the action choice seems to be influenced by how the objective is specified.

While not always present, at least one of these biases types was observed over the course of the presented experiments, which involved two classic games: *Prisoner's Dilemma* and *Stag Hunt*. Other key findings include the discovery that LLMs seem to be more affected by these types of biases than humans. Additionally, having tested with various prompting techniques, models do not seem to improve, suggesting that prompt engineering may not be the correct way to obtain better performances.

4.3 LLMs and The Prisoner's Dilemma

This last section is dedicated to a brief introduction of the findings of previous studies on LLMs in the context of prisoner's dilemma games. These served as the main inspiration for the experiments carried out during the second part of this dissertation.

4.3.1 Comparing Human and LLM players

The *Beyond Nash Equilibrium*[30] paper presents an in-depth comparison of human and LLM players in terms of *bounded rationality*. The expression refers to human's tendency to rationally deviate from the optimal immediate choice as a consequence of their *"cognitive limitations, perceptual filters or contextual factors"*. This would indicate that, to stride towards the simulation of human behaviour, deducing and pursuing the Nash equilibrium is not the only prerequisite. A clear example reported by the authors is related to the prisoner's dilemma, where humans tend to cooperate even in one-shot games because they still consider the possibility of a future meeting. Using experiments modelled closely after those created for humans it was shown that LLMs display a similar, albeit exaggerated version of human's bounded rationality. Examples of this behaviour emerge by comparison: where humans tend to adopt flexible strategies, LLMs tend to adopt one pattern and maintain it for the entire game. Additionally, LLMs show clear signs of weaker modelling capabilities of environment and opponents which leads to the adoption of inadequate strategies or to a lack of reaction against certain players. Lastly, it was observed that different families of models seemed to show some distinctive "fingerprints" in the way they play. This would suggest that training corpora and methodologies may affect the overall behaviour even more than model complexity.

4.3.2 LLMs in One Shot Games

The behaviour of LLMs in one-shot games of the prisoner's dilemma was investigated inside this publication[17]. During the course of this experiment, the objective was to analyse the choices of LLMs in a context with few learning opportunities, given the absence of repetition. Over the course of this large scale study, more than 350 treatments were deployed to analyse the behaviour of the agents over the widest range of conditions. To provide additional insight, some treatments involved asking the models what they expected their opponent to do. Many payoff configurations were explored, as well as condition mechanisms to push the agent to adopt a specific strategy. The results show that, independently of the model used, LLMs appeared to act more altruistically, cooperating in over 70% of the games. This is surprising, considering that humans have naturally a much lower cooperation rate in one-shot games, at around 30%. It is even more surprising considering that most agents seem to expect that other will defect, but choose nonetheless to cooperate. This phenomenon is interpreted as an inherent bias towards cooperation, potentially a result of model alignment towards positive behaviour. It was also observed that agents seemed to not notice payoffs variations, with their behaviour being way more dependent on context definition than the actual reward values. Further exams based on correlation seem to suggest that the overall behaviour of these agents seems to be extremely predictable, with high correlations between certain combinations of initialization variables and the end results.

4.3.3 LLMs in Iterated Games

Lastly, we will briefly present the *Nicer than humans*[9] paper which was the most important for the inception of the experiments of Part II. Here the authors investigate the behaviour of LLM agents over the course of multiple iterated prisoner's dilemma game. To do so they deployed metrics from the field of behavioural economics that monitor the strategic profile of the agent. The measures include *Niceness*, *Forgiveness*, and *Retaliation* which mirror the key strategic aspects defined in Chapter 2. The culmination of the strategy profile analysis is the usage of the SFEM (*Strategy Frequency Estimation Method*) which analyses the gameplay of an agent and assigns the closest existing strategy profile from a wide list of established ones. By observing the agent's behaviour, it was discovered that even in iterated games most LLMs have a tendency to act more cooperatively than humans (hence the title). Through the SFEM analysis it was also noted that models usually have nice strategies mostly aligned with *Grim Trigger* which tend to evolve into a straight up *Always Cooperate* over the course of longer games. It was however noted that some models, like Llama 3, were less inclined towards cooperation, which is surprising, considering that Llama 2 had the opposite behaviour. Another important focus of this study was addressing the comprehension capabilities of the subject models, this was achieved by questioning them directly over multiple aspects of the game. Aside some utterances concerning the payoff matrix, all models correctly guessed the answers, this was used as a validation of the models' understanding of the problem. Overall, this paper suggested a useful guideline on how to properly investigate the agent's behaviour, on top of providing another insight on how these agents approach game theoretic scenarios.

4.4 Thesis Contributions

While prior work investigates LLM strategic behaviour, little attention has been devoted to the architectural mediation of reasoning in iterated strategic settings. In particular, the comparative impact of tool-based augmentation versus internal multi-agent reflection remains underexplored. The experiments presented in Part II will provide a first insight into this field of investigation by comparing two separate architectures, each reflecting two paradigms to improve agent decision-making.

On one side there will be an example of tool-augmented agent which will use logic-driven approaches to aid its reasoning. On the other a more reflection based architecture models its thought process as a formal debate between two sub-agents, introducing a multi-persona negotiation with the objective of determining the right action. Conceptually, the two structures follow two ways in which the decision-making process could be improved, each with its own strengths and weaknesses. Instead of focusing on one single approach, this work will try to shed some light on these elements.

To investigate properly the capabilities of each architecture and determine their behaviours, they will be tested on Prisoner Dilemma games. As highlighted in previous sections, these games, while simple, offer a vast range of interpretative possibilities. Compared to previous studies, which focused primarily on the specific behaviour of single LLM agents, this thesis will delve into the response capabilities of larger structures. Taking inspiration from other works, we will probe the effects of different experimental settings. Overall, this projects inserts into the growing literature that explores the behaviour of LLM Agents. However, it tries to build upon it by introducing agents with deeper cognitive capabilities, modelled following the tool-calling and agent debate paradigms. By deploying classical experimental techniques from the field of social sciences and game theory, this study places itself in an interesting position. As A.I. systems become progressively more integrated in various decision-making scenarios, asserting their flaws and strong points in those sectors becomes essential. So, we will explore this line of research, providing a good starting point for future enquiries into this challenging territory.

Part II

Part II — A Comparative Study of Agent Architectures

Premises

This second part of the dissertation is dedicated to a series of experiments which will join all the concepts mentioned across Part I. Before starting, it is worth mentioning the context behind this work. As discussed in the introduction, this project stemmed from a collaboration with the University of Birmingham's Centre for Artificial Intelligence in Government which is currently developing a multi-agent simulation of the European Council. One of the current goals of the team is to expand the single agent's capabilities of simulating human behaviour. This is a necessity, since the negotiation process is complex and would require each agent to capture many of the aspects a human negotiator possesses. From this premise arose the idea of creating a more complex agent, which comprised multiple modules to capture different aspects of the negotiator. After brainstorming different ideas, two main directions for development emerged:

- Structure each agent as a multi-agent system, where modules with different competences debated.
- Create a single agent equipped with tools that provide the necessary components.

Given these premises, the main research question was defined: investigating how much each architecture managed to influence the behaviour of the agent. What was lacking was a specific context to set as the stage for the experiments. At this point, game theory emerged as the most suitable option. Given its central role in the dedicated literature and its overall simplicity, prisoner's dilemma was chosen as the main experimental setting. Combining these two aspects, we will define the research question for this thesis:

"Given two agent architectures: one based on agent debate and one on function calling, how do they affect the decision-making process of a player agent in the context of multiple prisoner's dilemma scenarios?"

Chapter 1

Modelling Game Theoretical Experiments

The presented experiments are the ultimate product of a build-up of previous investigations, mostly based on one-shot versions of the prisoner's dilemma. The initial tests were run for different reasons, among which there was the necessity to find the most appropriate framework to build the final system upon. Once enough confidence was gathered, it was time to properly sit down and design a solid and structured framework that could support the desired experiments. After some initial runs, it was decided to build a small, yet effective framework upon the available Ag2 library, previously seen in P. I, Ch. 3.

This chapter will be dedicated to explaining the overall architecture, as its development is a key part of the overall project.

1.1 Building a Framework for the Experiments

Coming from a software engineering background, the design of a system to support the proposed experiments followed specific requisites and principles. The architecture was to be designed to be as modular as possible, in order to be able to test each component separately and to allow the integration of new parts in the future. From the point of view of requisites, the goals of the frameworks were the following:

- Being able to define multiple games, each one with a specific set of payoff values.
- Accommodate for different types of player agents, from LLM based to Rule-based.
- Allow the usage of multiple prompting techniques for LLM players.
- Allow the definition of tools for the LLM agents to call to support their decision-making.
- Design the overall architecture to consent an easy variation of the settings without having to restructure the code.
- Account for the limitations of querying LLM APIs repeatedly.

Considering these requisites, a complete framework was designed, in the respect of the core principles of software engineering, such as the encapsulation principle. Overall, the software was written entirely in the Python language, considering its key support for most a.i. related programming. The main foundation libraries for the development was Ag2, an open-source branch of the original Autogen project. The basis is well established and support all the vast majority of available LLM APIs. As previously expressed, Ag2 is framework centred around the communication flow between agent which is structured around two elements: Conversable Agents and Chats. Ag2 was selected because its conversation abstraction allows fine-grained control over turn-taking dynamics. This is essential to test whether structured deliberation, rather than mere multi-agent presence, impacts strategic stability. On top of this, Ag2's ConversableAgent class is designed to work even in single-agent mode, which, combined to its tool-calling capabilities, make it the best choice for implementation. Another important component was the NashPy[13] library, which is specialized in the definition and manipulation of structures related to game theory.

1.2 The Framework's Architecture

Aiming to answer to the requisites presented in the previous section, the proposed framework's architecture is designed on the idea of multiple separate modules, each one defined in a separate file. The final goal would be to use the different components together in a single experiment main file, which can be set up for a specific run through a separate configuration file. There are a total of 6 separate modules which define the overall architecture, we will briefly present them all. With the objective of respecting the principle of interaction between interfaces, all modules present abstract classes versions of the implemented entities. The following descriptions will concern those, as the specific mechanisms of different implementations will be detailed, if necessary, in future sections. Figures 1.1 and 2.15 show a complete overview of how the different framework components interact.

1.2.1 Games and Rules — (*Games.py*)

This module contains two main abstractions which, together, help define the general framing of a game theoretic match between two or more players.

Rules. This abstraction encompasses the concepts of points assignment concerning the specifics of a game. A set of rules is defined by providing a series of payoff values inside the constructor of the class. Once they are passed, they are saved as an internal payoff matrix. This component is tasked with managing access to the payoff values through two main methods.

`get_payoffs` returns the payoff matrix to any external observer.

`compute_payoffs` accepts a series of strings indicating the actions of each player, validates them, and returns an ordered array of appropriate payoffs for each player.

Abstracting the concept of rules allows the automatization of the point assignment procedure during the experiments. This separation is designed to ease the investigation of different payoff distributions. Leaving the rest of the architecture unchanged detaching reward assignment enables experimenting with the classic Prisoner's Dilemma, Prisoner's Delight, and Prisoner's Confusion. Beyond the scope of the presented experiment, this design decision captures a fundamental abstraction of game theoretic experiments, which are always centred on payoff assignment. By giving this specific part its autonomy we stress the importance of these concepts as core tuning parameters for the entire experiment.

Game. This abstract class defined the main manager of the entire experiment, i.e. the game itself. It is expected to receive a series of rules inside its constructor, along with a list of all player agents that will take part in it. The main function of this unit will be keeping track of the interactions between players by calling their appropriate methods and handling the overall state of the game. This entire function is contained in the `play` method, which manages the flow of a game, prompting the agents with their next moves and registering the effects these have over the course of the match. Once the specified end status is reached, this method is also tasked with returning a summary of the game, in order to allow analysis. It may seem superfluous to underline the importance of a *Game* class in a framework designed for game theoretic experiments, but a few aspects introduced here are worth mentioning. This software component is the orchestrator of the activity all playing agents will take part in. In the context of the experiments that will follow, this structure creates an isolated container where the agent's interactions can run smoothly. Together with *Rules*, these two aspects encompass all the parameters that belong to the experiment, aside from the subjects. Being designed to integrate any kind of player make this abstraction modular, accentuating its reusability in future investigations.

1.2.2 Player Agents — (*Agents.py*) and (*LLMAgents.py*)

This module is entirely dedicated to the classes that act as agents inside each game. To avoid circular dependencies, it was decided to decouple this module by creating two separate files. One file manages the general abstraction for player agents and the concrete classes for Rule-Based agents, while the other is dedicated exclusively to all LLM-based ones.

PlayerAgent. The main abstraction of this module encompasses the concept of a player inside a game-theoretic setting. Each player can be assigned a name, an initial score and history from its constructor. The score represents the amount of points the player has gathered throughout the game, while the history is a list containing the sequence of actions each player chose. Functionally speaking, this class presents classical methods to access its internal status:

`getScore`, `setScore` and `updateScore` allow mediated access to the internal score variable, with the latter method allowing to pass the obtained reward and update the current value for the player.

`updateHistory` is used to pass an element describing the last turn of interaction between players, to allow a change in the internal representation of the game of the agent.

`generateResponse` is the main interaction method of the agent, as it will invoke the internal reasoning, independent of its implementation, to produce the response in the format required by the game. Generally speaking, it should be a string describing the action to be performed.

The player is the main subject of any game-theoretic experiment. As the literature on this topic suggests, the nature of the actor which produces a response can very greatly. It seems only natural to model this aspect of game theory independently of what process lays behind it. This supports the interaction between different kinds of agents, as required by the experiments. By wrapping agent behaviour with a proper interface we enabled the creation of games between generic PlayerAgents. In the presented case, an LLM-based agent and a rule-based one. As a design choice this expands the scope of other potential experiments, since the same structure can be easily repurposed. Potentially, this enables LLM vs LLM games, a new edition of Axelrod's tournament or the introduction of human players, through a dedicated input interface.

1.2.3 Rule-Based Strategies — (*Strategies.py*)

This module is dedicated to the definition of various strategy mechanisms for any rule-based agent. Since the framework is built to support games between agents with different natures, defining this concept seemed necessary as it represents the core guidance for a rule-based action's decision-making.

Strategy. This is the core abstraction of this module, it is quite simple and requires no further parameter inside its construction. `next_action` is the core method of this class, it expects the history of the game or a portion of it and uses it to compute a response following a specific set of criteria. This method is expected to be called inside the `generateResponse` of classical rule-based agents.

While the experiments will revolve around the behaviour analysis of LLM based agents, it is important to devote some attentions to their predecessors. Rule-based strategies have dominated the early stages of studies one autonomous agents and game theory. It was considered necessary to provide an abstraction that encapsulates the internal logic of such agents. Practically, this choice allows an easy switch between strategies for any rule-based agent involved in the game. It highlights again the attention towards modularity, since changing this element reshapes the entire functioning of an agent without modifying any of the other components.

1.2.4 Prompt Generators — (*Prompts.py*)

A fundamental component of the architecture and one at the core of its modular nature, as it allows for the definition of multiple prompting strategies for the same game. This module allows the definition of a series of units that generate appropriate prompts for every component of an experiment. This helps with the flexibility of the architecture, since it allows for the definition of multiple prompting units, each one dedicated to a setting or a specific prompting technique, without changing the rest of the code. The main parent abstract class would be **Prompter**, but given the fact the entire experiment hinges on the prisoner's dilemma a dedicated abstraction was defined extending it. This one will be the subject of the description.

PD_Prompter. This abstract class is tasked with handling the generation of all prompts concerning to a specific suit of experiments. Given its nature, it contains a wide range of methods, mostly similar between each other but tailored to the specific component they are referring to. We will describe one of them to give an idea of its mechanisms. This class accepts as input from the constructor a Rules object, whose contents will be used to build the appropriate prompts by specifying the payoffs when needed.

`generatePromptDecisionMaker`, as the name suggest, is tasked with providing the caller with an appropriate prompt for the initialization of the decision maker unit, whether it be inserted inside a larger architecture or a simple LLM. The method is quite simple, accepting a few parameters specific to the experiment, such as the number of rounds of play.

As discussed previously, when dealing with LLM systems, prompting is one of, if not the most, crucial aspect to consider. It was thus deemed necessary to gather every aspect of this procedure inside one single unit. Having a dedicated set of classes responsible for this activity centralizes the management of such critical aspects. Additionally, again for the sake of reusability, this allows an easy change in the overall context of the experiment. The Prisoner's Dilemma can be presented in a multitude of scenarios, all adopting the same basic rules. With this abstraction we create an effective presentation environment for the agents. Switching the prompter unit without modifying the rules will result in the same game but presented differently. If the agents are meant to be players in a game, this unit is tasked with describing its elements to them through text. Depending on the phrasing of the descriptions, the range of generated responses may vary greatly.

1.2.5 Tools — (*Tools.py*)

As a counterpart to the strategies' module, this one is used to define the concept of agent tools. Here we have one of the core functionalities of LLM-based agents, as these represent their main access to function calling and, as such, require some precise specifications.

Tool. This abstraction represents a tool available at runtime during the course of the game. It presents mostly class attributes, which define its work settings. Each agent possessed a single tool of any given type. The main requirement to the creation of a tool is the passage of an appropriate prompt generator from the dedicated prompt module. This is important because it will be used for the generation of the tool description and to frame the output in the context of the appropriate scenario. As stated before, the tool description is essential for the agent to understand its purpose and invoke it properly when necessary.

`get_tool_result` is the heart of this abstraction, as it is a static method tasked with generating the necessary output for each tool. The method must be defined carefully as it will require an accurate description of the specific input parameters. Once correctly invoked, it will be executed and return the result in a text-friendly format, according to the specifications of the prompting unit.

Tools represent one of the two key features the agent under investigation will use to augment their capabilities. The decision to dedicate a module of the software to them is thus obvious. There was a need to provide a higher level of customizability to such a crucial element of the framework. It wraps the Autogen component into a dedicated unit that includes the view defined by the prompt. A tool provides the agent with an external computation module which introduces symbolic knowledge into the LLM sub-symbolic reasoning. This shifts the architecture towards a neuro-symbolic model, by giving the agent a logic-driven payoff assessment method. This addition is expected to counter the bounded rationality effects that previous literature detected in LLM behaviour.

1.2.6 Request Limitation Management — (*Sleeper.py*)

Less related to the structure of the game and more useful for the practical aspect of running the experiments when an LLM is involved, this model is tasked with the management of token quotas. When running experiments that involve LLM API calls, it must always be considered that there are limits to how many calls a user can make over the course of a minute, an hour, or a day. The same can be said about the number of tokens generated for the responses. To avoid unnecessary breaks and exceptions, a system to register consumption was developed. The **Sleeper** class is created by specifying a maximal amount of tokens and calls that can be made in different intervals across a day. Then, during its construction, the module will load the current usage from a separate `quotas.env` file and reset some of its entries if enough time elapsed from the last call.

`update_usage` is important for the update of the internal state of the sleeper, since it takes as input the number of calls and tokens consumed by the previous operation (returned by the `generateResponse` methods) and updates the current usage quotas.

`check_and_sleep` is the main method of this class, and it manages the suspension mechanism to avoid exceeding the limitations. It is supposed to be called after `update_usage`. This method checks if the current quotas were exceeded and, if necessary, suspends the execution until enough time has passed for the limit to be removed.

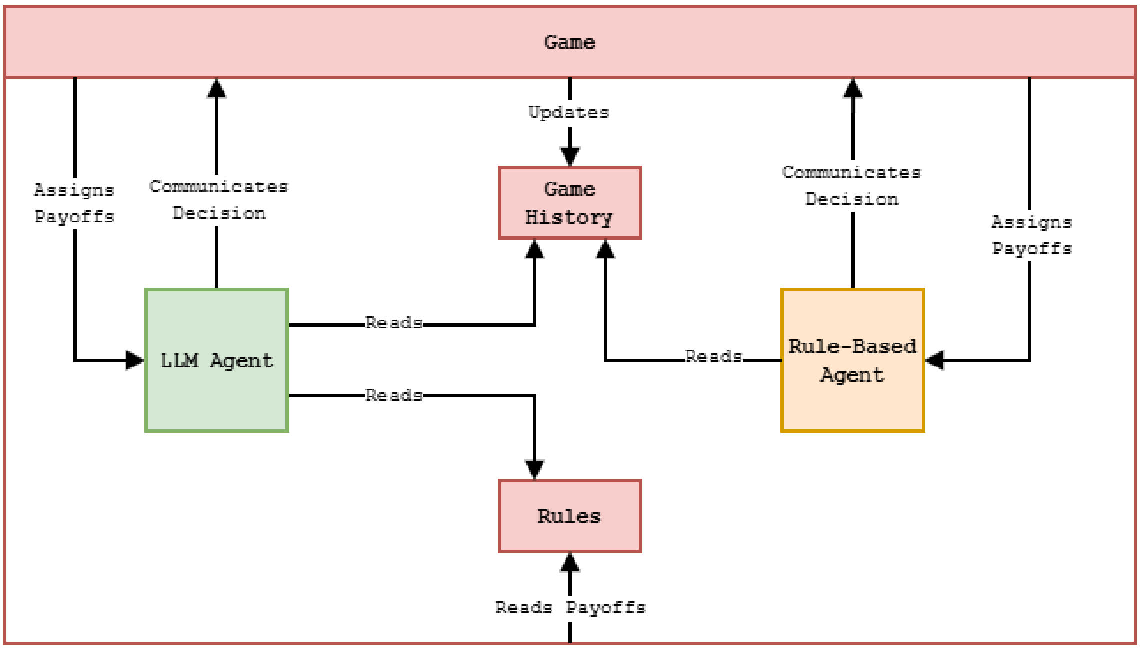


Figure 1.1: Diagram of Game and Players

Chapter 2

Agent architectures under analysis and methodologies

2.1 Experimental Setup

Now the all the necessary software components have been properly defined, it is finally time to enter in the heart of the experiment. As expressed previously, the objective is to investigate the behaviour of differently structured LLM-based agents under in different settings related to the prisoner's dilemma. The idea of creating multiple situations comes from the necessity of investigating how much of the overall context the agents seem to comprehend and how they react to different variations of the scenario they are in. For example: are they capable of recognizing a that different equilibrium conditions will require a change in strategy? Will the presence of an exploitable opponent lead to a different approach? Most of all: which is the impact of each architecture in this decision-making process? To properly investigate a wide range of scenarios it was decided to work along four main factors and run experiments under every possible combination of them. Each of these factors will be described below, with their specific setting, the meaning behind their choice and the expected effect they may have. Independently of the setting the experiment will always involve an LLM-based agent playing against a fixed opponent with a rule-based strategy. The two agents will be involved in a series of iterated prisoner's dilemma adjacent games. During the course of each game, both agents will choose between to options: *cooperate* or *defect*. Both agents will have full access to the current history of the game up to that point, consisting of all players' previous actions. Additionally, to gain further insight in the inner reasoning of the LLM-based agent, after each round a series of questions will be asked about various aspects of the game. The game history is always presented in the format described below in Figure 2.1

```
ADDITIONAL INFORMATION:

Summary of the last <N_PASSED_ROUNDS> interrogation rounds:
Round 1 - Suspect chose: <AGENT_CHOICE_1>, Suspect's partner chose: <OPPONENT_CHOICE_1>
Round 2 - Suspect chose: <AGENT_CHOICE_2>, Suspect's partner chose: <OPPONENT_CHOICE_2>
Round 3 - Suspect chose: <AGENT_CHOICE_3>, Suspect's partner chose: <OPPONENT_CHOICE_3>
```

Figure 2.1: Example of game history at round 3.

2.1.1 Optimal Choice

One of the first things to verify when considering the decision-making capabilities of an agent is to verify whether it is capable of discerning different setting it may find itself in. In the context of a game theoretic experiment, this can be clearly observed by noticing how the strategy profile changes if the payoffs are shifted with varying degrees of intensity. Strategy profiles are highly dependent from the payoff matrices and, in the context of the prisoner's dilemma, specifically in the one-shot version. The Nash equilibrium is quite easy to compute. While initial studies were conducted by exploring multiple configurations of payoff matrices, it was quickly observed that there are three main situations available which can affect the decision making of the agent.

Generally speaking, the three mentioned settings represent three separate payoff matrices which mirror the ones presented in P. I, Ch. 2. In order to enhance the difference between the various payoffs and make them more noticeable for any potential LLM agent, it was decided to set the parameters using permutations of $[0, 5, 10, 15]$. This way the reciprocal ratio between payoffs would be kept similar to the classic formulation, accentuating the distances.

Prisoner's Dilemma. Analogous to the classical setting, in this case we have:

$$T = 15, R = 10, P = 5, S = 0$$

In this situation, *defect* is the safest choice in a one-shot game, while *cooperate* may be identified as the best choice over the course of multiple games.

Prisoner's Delight. To obtain a completely reversed setting, it was simply necessary to invert the values, ending up with:

$$T = 0, R = 5, P = 10, S = 15$$

While this scenario may appear nonsensical from the point of view of the original game, it represents a setup where the best choice in a one-shot game is *cooperate*, while *defect* may become the optimal choice in repeated games. The objective of this setting is to test the agent's capability of spotting the optimal choice even when presented in a more unclear way.

Prisoner's Confusion. This setting represents more of a control case to contrast with the other two since it involves a configuration with:

$$T = 10, R = 15, P = 5, S = 0$$

In this case, in a one shot game, there is no safest choice, so the agent may find itself confused on what to do. With this condition being more neutral, it could provide a way to highlight potential bias towards the two options as consequence of all other factors. It is necessary to mention, at this point, a specific detail of how things were implemented in the presentation of the problem in the context of all agents. Given how famous the prisoner's dilemma is as a setting, there was a high chance for the LLM agents to recognize it as a game and act based on that premise. To avoid this, it was decided to present the situation in a more grounded setting, describing it through its proper scenario rather than a game. This meant that all payoffs, which in the context of the setting represent years in prison, had to be "reversed" for it to make sense. This way high payoff was associated with a low sentence. On a mathematical level, this was simply achieved by subtracting the payoff from the maximum value.

$$T_{year} = max(T, R, P, S) - T$$

As an example, if $T = 15$, in the context of the new framing it was presented as being the lowest possible sentence, i.e. $15 - T = 0$ years in prison.

2.1.2 Suggested Decision

The next setting examined revolved around the capacity of conditioning the overall system towards a specific decision, a parameter which we will refer to as *attitude*. The attitude represents a proposed behaviour that the assisting unit will present to the main decision maker agent. This usually translates into a specific action to be presented as the best, regardless of it actually being true. The specifics on how each attitude is injected into the behaviour of each agent will be explained in the part dedicated to their architectures. Attitudes are meant to elicit a confrontation with the scenario the agent is currently facing. The idea is that whenever scenario and attitude are aligned, the agent should be strongly inclined towards that one choice (whether *cooperate* or *defect*). On the other hand, when the attitude and scenario are not aligned, the agent should behave under more uncertainty. From a technical point, the experiments were carried out using three attitudes:

- Cooperate, the agent is prompted to cooperate with its partner
- Defect, the agent's suggestion is to defect against its partner

- Compute, the agent is invited to analyse the current scenario and deduce by itself the best option

2.1.3 Number of Rounds

A key observation in previous experiments was that agents tend to act differently in games with different duration. To address this phenomenon, it was decided to run two separate versions of the IPD. The general expectation is that a longer game would see the rise of a more clear orientation towards the Pareto optimal strategy profile. As a consequence, a choice was made to carry out the experiment with 10 and 20 round games. In both cases, the agents were properly informed of the duration of the game. Another thing we may expect to notice, especially in the case of the classic PD, is the "last round defection", which represents a common occurrence in many experiments. The reason for this is simple: even if two agents have learned to cooperate together, recognizing that a round will be the last, they might be more tempted to opt for defection, assuming there will not be other interactions later. Reasoning by induction, this could be extended to all previous rounds. Considering this, we may expect the last rounds to contain more defection.

2.1.4 Type of Opponent

The last setting variation introduced in the experiments is the strategy of the opponent. As described above, the games will always see an LLM-agent against a rule-based one. This choice was intentional, because it allows to analyse how the first may react in case it recognizes a specific pattern in the second's choices. For example, if the agent understands that the opponent is always choosing *cooperate*, it may decide to exploit this "weakness" and defecting more often. A more flexible approach to the analysis of the opponent's strategy will be considered the sign of a higher reasoning capability. As seen before, playing by considering the other players' choices is a requisite for a level-k type of thought, according to Nagel's classification. Considering these premises, three strategies were deployed to define the opponents among those described in P. I, Ch. 2:

- Always Cooperate, with the expectation of seeing an increase in defection to exploit it.
- Always Defect, expecting a defection to avoid being exploited.
- Tit for Tat, potentially eliciting cooperation, given its retaliatory nature.

2.1.5 Models under examination

While not directly considered one of the aspects under examination, since it resides outside the scope of the game itself, a few words must be spent discussing the LLM models used during the experiments. Given that one of the agents requires a proper LLM to work, different solutions were evaluated before making the final choice. Many factors were taken into account in the choice of which models to use for carrying out the experiments. One key aspect was the presence of a function calling capability, which is not guaranteed by most basic models. Additionally, affordability was another deciding aspect: given the necessity of generating a vast number of tokens, especially in the case of the debating agent, models with a lower cost were prioritized. In the end, the experiments were run using two main LLM APIs:

`gpt-4o-mini`. This model is a reliable staple of LLM experiments, representing a good compromise between complexity and cost-efficiency. From the point of view of the experiments it was chosen a representative of closed-source models. Compared to the other model, it is larger and more properly instruction tuned, leveraging a higher parameter space.

`ministral-8b`. This lightweight model was deployed for its speed and overall reliability despite its small size. It is an example of open-source model but shows nonetheless promising aspects, such as its reliable tool-calling capability, and its surprising accuracy.

With the objective of studying the model's behaviour in a "natural" setting, the basic temperature for each LLM was not set, leaving the basic configuration.

2.2 Evaluating the reasoning

In addition to evaluating the behaviour of the agents, another important aspect is to investigate how they reason throughout the experiments. Furthermore, it is essential to constantly assess if they are properly understanding the task they are currently facing or if they are randomly outputting values. To investigate this, it was decided to add another layer to experiments, by asking directly the agents to answer questions between rounds. On top of giving insight into their reasoning, this practice may have a positive effect on stimulating their thought process, thus improving further decisions. Considering the computational cost of generating these kinds of answers, especially for the debate-based model, it was decided to limit their number to 4, carefully selecting them to assess different aspects of the deliberation. The nature of each question and the reasons behind their insertion will be explained below. It must be noted that, just like before asking for a decision, a summary of all previous interactions during the game was provided, to elicit a more informed response. This information is always presented as in Figure 2.1.

2.2.1 Opponent Intention Prediction

```
What is going to be your partner's next move?
(Express the answer as ANSWER:<ANSWER>)
```

The first question tackles the capability of the agent of analysing the opponent's next moves, it is meant to elicit reasoning on the previous interactions in an attempt to figure out a pattern. The results from this question can be compared with the objective outputs of the opponent's next move to see how often the agent managed to correctly guess. Anticipating the opponent's next move is a key ability to assess the awareness of the system in the context of the overall problem. It tests its reasoning and allows confrontations with its next choices. For example, it could be interesting to see if the agent still opts to cooperate even when anticipating a defection.

2.2.2 Understanding of immediate information

```
How many rounds have you gone through so far?
(Express the answer as ANSWER:<N_ROUNDS>)
```

The second question seems apparently redundant, given that the number of elapsed rounds is expressed directly in the additional information. However, the storage of this data from any model is not always guaranteed and having a control question such as this can help to investigate the confusion status of the agent. If an obvious question such as this does not receive a correct answer, we can assume that the model is struggling with the information at its disposal. This could be used as a red flag for the model's overall poor reasoning performances.

2.2.3 Memory of initial context

```
Which are the maximum and minimum number of years you may get
from this round?
(Express the answer as ANSWER: MAXIMUM:<MAX>, MINUMUM:<MIN>)
```

The third question is meant to test the agent's long term memory capabilities. Since the information about the payoffs is specified in the initial prompt, asking it to provide this information may help it to recall the current stakes. Given the limited size of context windows for these kinds of models, its capability to answer this question may degrade with each passing round.

2.2.4 Arithmetical computation

```
How many years of sentence have you received so far?
(Express the answer as ANSWER:<TOTAL_YEARS>)
```

This last question is meant to evaluate the agent's overall reasoning starting from available information. By combining the available history with the memory of the payoffs, it is possible to compute the required number. The task is hard, considering most LLMs are not well suited for arithmetical computations. Nonetheless, this question is expected to receive an abundance of wrong answers, but the extent of the errors will be a deciding factor in determining which of the tested architecture performs better in this specific subject.

2.3 Architectures under comparison

Now, we move to the core subjects of the overall experiments: the architectures. As discussed before, the objective is to evaluate which of the two proposed structures is the most effective at eliciting reasoning and, at the same time, suggest a specific course of action to an LLM-based agent. To provide an impartial baseline, each of the two agent setups will be compared to a plain LLM, prompted to answer similarly. This will additionally provide a case for whether a more complex structure, as those designed are, is necessary at all. Each architecture will be presented in terms of its inner workings and, given how each of them contains one or more LLM-based units, the specifics of their initialization. Descriptions will include the initial prompts for each of their components and those deployed for the generation of their responses.

2.3.1 Plain LLM Agent

This architecture represents the baseline against which all others will be compared. It is a plain LLM agent, without any tool available and initialized by describing the problem scenario. The role prompt for a Plain LLM agent is displayed in Figure 2.2:

```
You are a person.
SCENARIO:
You and your partner were arrested and are waiting separately for interrogation.
You will be interrogated for <N_ROUNDS> rounds.
During each interrogation round, you and your partner will independently and simultaneously choose one action: 'Cooperate' or 'Defect'.
Depending on your choices, you will receive different sentences.
If you choose 'Defect' and your partner chooses 'Defect', you will receive <P_YEARS> years in prison and you partner will receive <P_YEARS> years in prison.
If you choose 'Defect' and your partner chooses 'Cooperate', you will receive <T_YEARS> years in prison and you partner will receive <S_YEARS> years in prison.
If you choose 'Cooperate' and your partner chooses 'Defect', you will receive <S_YEARS> years in prison and you partner will receive <T_YEARS> years in prison.
If you choose 'Cooperate' and your partner chooses 'Cooperate', you will receive <R_YEARS> years in prison and you partner will receive <R_YEARS> years in prison.
Your objective is to minimise your total years in prison over all interrogation rounds.
```

Figure 2.2: Role Prompt for Plain LLM agent.

During the course of each round of the game, the agent was prompted to express its decision in JSON format, to standardize answers. Additionally, a line was added to induce the agent to reflect on the question. Several studies suggested that doing so enhances the reasoning capabilities. As specified above, whenever a game history was available, it was added after the prompt in the format specified above. Considering this model has no way to inject an external influence, the *attitude* is not present in the overall prompt. This choice was considered reasonable, since this is supposed to be a benchmark for the two main architectures. A full rendition of the prompt used to obtain the decision can be seen in Figure 2.3

```
Think about this carefully.


OUTPUT FORMAT:
Once you have made your final decision, respond with ONLY this JSON format and nothing else:
{'DECISION': '<YOUR_DECISION>'}
```

Figure 2.3: Decision Prompt for Plain LLM agent.

```
Your task is to answer this question:
<QUESTION>
Think about this carefully.


OUTPUT FORMAT:
Once you are ready to provide an answer, respond with ONLY this JSON format and nothing else:
{'ANSWER': '<YOUR_ANSWER>'}
```

Figure 2.4: Question Prompt for Plain LLM agent.

The prompt used to generate an answer to each of the questions is structured similarly (Figure 2.4).

2.3.2 Tool-Based Agent

Here we introduce the first of the two architectures under examination. The structure is quite simple as it is based on an agentic LLM equipped with a single tool to aid in the decision. The main introduction of this architecture is a tool, a software component which allows the agent to implement a simple function which can help to guide its thought process. Over the course of the experiments, two main tool objects were deployed and experimented upon: one handled the direct suggestions and the other implemented a more elaborate evaluation. However, it must be noted that the agent was always equipped with just one tool at a time. As highlighted before, the description of a tool and its function is essential to its usage by the agent. Even if potentially useful, a tool may not be chosen because its described function does not appear to match with the context. Considering that the chosen setting involved a realistic prisoner's dilemma scenario, it was decided to present the tool, whatever its implementation was, as a real world object the agent could interact with. Specifically, given it is suspected criminals we are dealing with, the design choice for the framing of the tool was a telephone with a direct line with a lawyer. The two tools implemented present a similar output but differ significantly in the way they are called. The *attitude* parameter determined which tool was assigned to the agent at the beginning of each run. In the context of the experiment, we will refer to the tool as `Fixed` with the attitudes *cooperate* and *defect*, while for *compute* we have a tool of type `Nash`. On the implementation side, a `Fixed` tool expects no input parameters and returns directly the *attitude* under the appropriate format. Conversely, a `Nash` tool requires from the agent to specify the four payoff values, then computes the best course of action using the `Nash.py` library given the payoffs provided, and returns a formatted version. In this second case, the agent is invited to reason on its situation, choosing the correct values to call the tool. To account for setting where there is no clear decision it was decided to use random choice. The tool definitions for `Fixed` and `Nash` can be seen respectively in Figures 2.5 and 2.6.

```
This tool is a phone to call your lawyer for advice during an interrogation.
"Input: There is no input information to provide.
Output: A brief suggestion from your lawyer.

IMPORTANT CONSTRAINTS:
- Call this tool ONLY ONCE per user message
- After receiving the lawyer's advice, you MUST proceed directly to making your decision
- DO NOT call this tool again unless the user asks a new question
- If you've already called this tool in this response, you already have the advice you need
```

Figure 2.5: Description of the tool (Fixed).

```
This tool is a phone to call your lawyer for advice during an interrogation.
Input: Information about the possible outcomes you're facing.
Output: A brief suggestion from your lawyer.

IMPORTANT CONSTRAINTS:
- Call this tool ONLY ONCE per user message
- After receiving the lawyer's advice, you MUST proceed directly to making your decision
- DO NOT call this tool again unless the user asks a new question
- If you've already called this tool in this response, you already have the advice you need
```

Figure 2.6: Description of the tool (Nash).

Since parameters need to be specified for the `Nash` tool, proper descriptions had to be specified even for them.

- temptation - `The number of years you get if you defect and your partner cooperates`
- reward - `The number of years you get if you cooperate and your partner cooperates`
- punishment - `The number of years you get if you defect and your partner defects`
- sucker - `The number of years you get if you cooperate and your partner defects`

The outputs for each tool were framed in the same way, in both cases it was necessary to specify the finality of its deliberation, to avoid repeated calls. This can be seen in Figure 2.7.

```
=== LAWYER CONSULTATION COMPLETE ===
Your lawyer advises: <ATTITUDE>
Reasoning: <ATTITUDE> is the best option.
This consultation is now finished. Based on this advice, make your decision and output it in the required JSON format immediately.
No further tool calls are needed or permitted.
```

Figure 2.7: Output format for the tool.

Concerning the agent itself, the role, decision, and question prompts are similar to those used for the plain LLM, for internal coherence. A few additional lines were added to indicate the availability of a tool to help the decision. They are shown in Figures 2.8, 2.9 and 2.10 respectively.

```
You are a person.
SCENARIO:
You and your partner were arrested and are waiting separately for interrogation.
You will be interrogated for <N_ROUNDS> rounds.
During each interrogation round, you and your partner will independently and simultaneously choose one action: 'Cooperate' or 'Defect'.
Depending on your choices, you will receive different sentences.
If you choose 'Defect' and your partner chooses 'Defect', you will receive <P_YEARS> years in prison and you partner will receive <P_YEARS> years in prison.
If you choose 'Defect' and your partner chooses 'Cooperate', you will receive <T_YEARS> years in prison and you partner will receive <S_YEARS> years in prison.
If you choose 'Cooperate' and your partner chooses 'Defect', you will receive <S_YEARS> years in prison and you partner will receive <T_YEARS> years in prison.
If you choose 'Cooperate' and your partner chooses 'Cooperate', you will receive <R_YEARS> years in prison and you partner will receive <R_YEARS> years in prison.
You have a tool at your disposal: a phone to call your lawyer.
Your objective is to minimise your total years in prison over all interrogation rounds.

CRITICAL RULES:
Use each tool only once.
DO NOT CALL AGAIN AFTER USING THE TOOL.
DO NOT CALL THE TOOL MULTIPLE TIMES BEFORE ANSWERING.
```

Figure 2.8: Role Prompt for tool-based agent.

```
Think about this carefully.

TOOL USAGE RULES:
- You may call your lawyer tool ONCE if you need advice
- After receiving advice (or if you don't need it), proceed directly to your decision
- DO NOT call any tools after making your decision

OUTPUT FORMAT:
Once you have made your final decision, respond with ONLY this JSON format and nothing else:
{'DECISION': '<YOUR_DECISION>'}
```

Figure 2.9: Decision Prompt for tool-based agent.

```
Your task is to answer this question:
<QUESTION>
Think about this carefully.

TOOL USAGE RULES:
- You may call your lawyer tool ONCE if you need advice
- After receiving advice (or if you don't need it), proceed directly to your decision
- DO NOT call any tools after making your decision

OUTPUT FORMAT:
Once you are ready to provide an answer, respond with ONLY this JSON format and nothing else:
{'ANSWER': '<YOUR_ANSWER>'}
```

Figure 2.10: Question Prompt for tool-based agent.

Concerning this architecture, there are a few additional considerations to do about its inception. Tool calling represents one of the first examples of integration between symbolic and sub-symbolic reasoning, and it is thus one of the most logical choices for model enhancement. Integrating an external reasoning systems into an LLM means delegating parts of probabilistic processing to an explainable external unit. The main challenge with this approach is describing correctly the tool to make sure the agent calls it properly. As LLMs are known for their occasional inconsistency in decision-making, especially when their temperature is not set, adding a grounded reference is expected to increase their analytical capabilities. To investigate this behaviour, the comparison between the effect of the different attitudes will be crucial. Particular attention will be given to the joint effect with the payoff structure. In this context, *Fixed* tools will help to determine how strong is the influence of the tool on the overall decision-making. We will track how strongly the tool attitude can push towards a decision or another regardless of what the best option is. Considering the *Nash* tool, on the other hand, the focus will be on how often it successfully guides the agent towards the correct decision. Given that the suggestion is based on the one-shot best decision, it is also expected a conditioning on the long term behaviour during an iterated game. Based on these intuitions, we can formulate some hypotheses concerning the comparison with the plain LLM agent.

HT1: Agents that use the *Fixed* tool will have higher suboptimal behaviour when the suggestion contrast the real best choice.

HT2: Agents using the *Nash* tool will express higher Nash-consistency, opting more frequently for the one-shot best decision.

HT3: Agents using the *Nash* tool will exhibit more attention towards payoff values, reducing the insensitivity observed in previous literature.

2.3.3 Debate-Based Agent

The second architecture presented is instead more articulated and consists of two similar, yet complementary parts. In this case, instead of having an external system to assist in the decision, the entire agent is constituted by two separate, smaller agents, which debate and analyse the situation together. To achieve this result, two agents with separate roles had to be defined, as well as a place to allow for their interaction. The first agent is the decision maker (i.e. the suspect), it is initialized by describing its general situation but without knowledge of the actual payoffs. It is tasked with providing the final decision. The second agent is the assistant (i.e. the lawyer), which has knowledge of the entire payoff matrix but is not allowed to provide the final decision. Whenever necessary, the decision maker agent is prompted to start a conversation with the assistant. The conversation is initialized by a specific prompt and is carried out following the round-robin pattern offered by the Ag2 library. The conversation is scheduled to go on until a specific keyword is generated, either `DECISION` or `ANSWER`, depending on the task. The keywords signify the generation of a proper response, which is extracted from the last message and processed. The idea behind this architecture is to elicit a discussion and a reflection between the two agents, which may push the two to a common resolution. To further incentivize this vision, the knowledge of the full problem, including the payoffs, was given to the agent which can not respond directly. This way, some form of knowledge sharing is necessary to make a decision. On the other hand, the decision maker knows how many rounds the game will last. The initialization process for the chat and the agents is particularly careful, as the conversation needs to respect a series of criteria.

- The decision-maker can not end the debate immediately.
- None of the agents can simulate the response of another one (a phenomenon known as role-play).
- A decision must be provided in a reasonable amount of rounds.

Concerning the influence of the *attitude* parameter, this is another aspect handled by the assistant agent, which is prompted to advise the other in this regard. Specifically, in the case of *cooperate* and *defect* it is invited to suggest those actions directly, while in the case of *compute* it will advise to “make the best choice”. Below, the initialization prompts for decision maker and assistant can be seen in Figures 2.13 and 2.12 respectively.

```
You are a person.

SCENARIO:
You and your partner were arrested and are waiting separately for interrogation.
You will be interrogated for <N_ROUNDS> rounds.
During each interrogation round, you and your partner will independentely and simultaneously choose one action.
Depending on your choices, you will receive different sentences.
You will be assisted by a lawyer during your interrogation.
You will debate with your lawyer to determine the best decision.
Your objective is to minimise your total years in prison over all interrogation rounds.

CRITICAL RULES:
You must debate with the lawyer at least once.
Do not simulate future rounds of the debate.
Do not simulate your lawyer's response.

EXAMPLE OF CORRECT RESPONSE:
I may agree with your point, but ...

EXAMPLE OF WRONG RESPONSE:
**Criminal**: I want to do this and this.
**Lawyer**: No, you should do that.
```

Figure 2.11: Role prompt for the decision maker agent.

```
You are a lawyer.

SCENARIO:
Your client is being interrogated and must make a decision.
Depending on their choices, you will receive the following sentences:
If they choose 'Defect' and their partner chooses 'Defect', they will receive <P_YEARS> years in prison and their partner will receive <P_YEARS> years in prison.
If they choose 'Defect' and their partner chooses 'Cooperate', they will receive <T_YEARS> years in prison and their partner will receive <S_YEARS> years in prison.
If they choose 'Cooperate' and their partner chooses 'Defect', they will receive <S_YEARS> years in prison and their partner will receive <T_YEARS> years in prison.
If they choose 'Cooperate' and their partner chooses 'Cooperate', they will receive <R_YEARS> years in prison and their partner will receive <R_YEARS> years in prison.
You should advise your client to choose <ATTITUDE>.
You will debate with your client to determine the best decision.
Your objective is to minimise your client's total years in prison over all interrogation rounds.

CRITICAL RULES:
You must debate with your client at least once
Do not simulate future rounds of the debate.
Do not simulate your client's response.

EXAMPLE OF CORRECT RESPONSE:
As your lawyer, I would suggest you to...

EXAMPLE OF WRONG RESPONSE:
**Criminal**: I want to do this and this.
**Lawyer**: No, you should do that.
```

Figure 2.12: Role prompt for the assistant agent

As specified, the chats were initialized by specifying their main purpose and reiterating the core rules of the debate. This choice was made since early experiments showed that agents violated rules the longer the game lasted. Below are shown the prompts for decision (Figure 2.13) and question (Figure 2.14) chats.

```
The objective of the debate is to determine the best decision for me.
We both need to provide arguments to support our own opinion.

CRITICAL DEBATE RULES:
We must express only our opinion.
The suspect can't simulate the responses of the lawyer.
The lawyer can't simulate the responses of the suspect.
Each response can only contain its proponent's own opinion.
Debaters are forbidden from simulating future responses.
Our opinions should be expressed in at most three sentences
Once the debate is concluded, I must express my final decision in the following JSON format:
{'DECISION': '<YOUR_DECISION>'}
```

Figure 2.13: Prompt to initialize the chat for the decision.

```
This is a debate between two agents: a suspect and its lawyer.
The objective of the debate is to determine the answer to this question:
<QUESTION>
Both agents need to provide arguments to support their own opinion

CRITICAL DEBATE RULES:
Both agents must express only their opinion.
The suspect can't simulate the responses of the lawyer.
The lawyer can't simulate the responses of the suspect.
Each response can only contain its proponent's own opinion.
Debaters are forbidden from simulating future responses.
Once the debate is concluded, the suspect must express their final decision in the following JSON format:
{'ANSWER': '<YOUR_ANSWER>'}
```

Figure 2.14: Prompt to initialize the chat for the answer.

The structure of this player agent reflects an attempt at imbuing it with a form of cognitive pluralism. Instead of viewing the agent as a single monolithic LLM, two separate personas are tasked with separate pieces of information and have to converge on a solution. The two personas are supposed to represent two separate "voices" inside the agent's "head", each with its unique point of view. On one side the *suspect* is the one which can make the decision but directly affected by it, while the *lawyer* can analyse in a more detached way, without choosing outright. The debate between the two agents should help a form of decomposition of the overall problem, through the natural flow of a discussion. This form of augmentation is more subtle than tool-calling and, compared to it, may show less accuracy in determining what is best. However, given its focus on deliberation, it may incentivize a more structured analysis of the situation, which results in a more complex modelling of the game development. The expectation is a reduction in impulsive biases in favour of a more complex approach, motivated by its deeper cognition. Concerning the experiment, it is expected a weaker stance when it comes to adherence to the *attitude*, but a more dynamic strategic approach. Based on these premises, we may formulate a series of hypotheses on debate-based agents.

HD1: Debate-based agents will exhibit more complex strategic profiles, especially in longer games.

HD2: Debate-based agents will show more accurate modelling capabilities, anticipating more accurately the opponents.

HD3: Debate-based agents will be less effected by the *attitude* than tool-based ones, but more focused on opponent strategy.

2.4 Result formats and elaboration.

2.4.1 Experiment result format

Having presented all the different settings that were tested during the experiments, it is necessary to explain which data they produced how they were processed. The main output of one run of experiments is structured to capture all the necessary outputs produced by the it in three `.csv` files, one for each payoff configuration. Specifically each file contains a total of 15 columns.

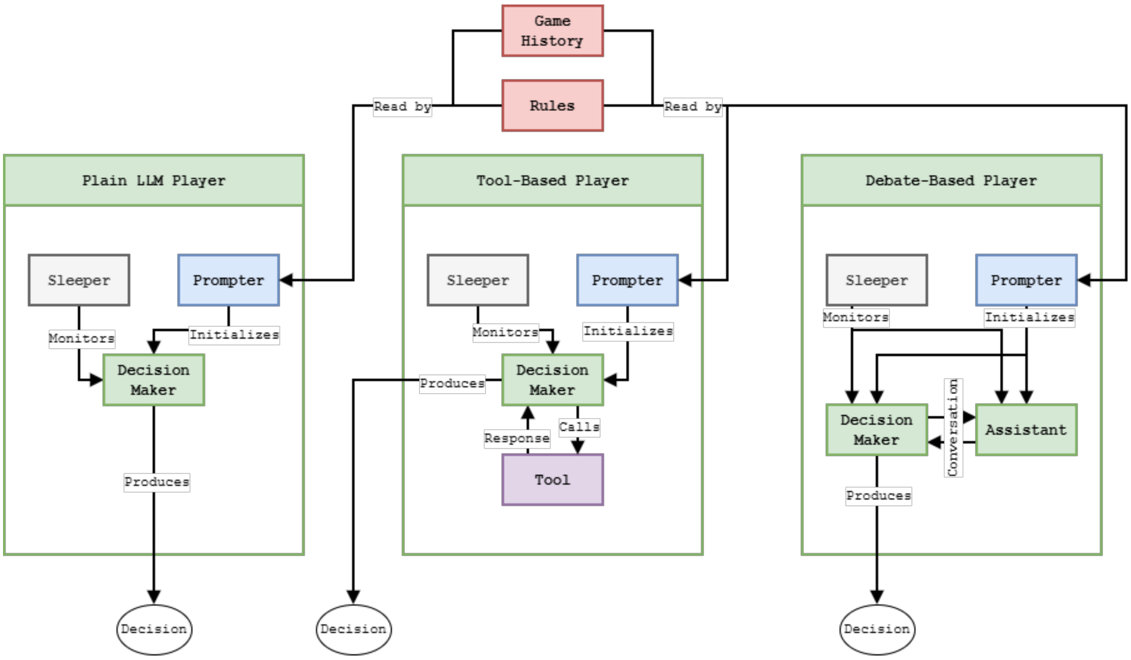


Figure 2.15: Diagram of Player Agent Structures

- 1 column with the round number.
- 2 columns with action choices for the two players.
- 2 columns with the scores of each player up to that round.
- 4 columns with the answers to the questions.
- 5 columns with the current settings for the experiment (attitude, number of rounds, architecture, opponent strategy, scenario[1] and model name)

Additionally, the results were organized through a tree of directories which branches based on the parameter choices, allowing for a structured division of the experiments. Once all output files were produced, they were all processed to generate a new `.csv` file containing information extracted from every game.

[1]This parameter would allow setting a prompting scenario different from the classic one, but was not used in this dissertation.

2.4.2 Game choice aggregation

Once all the experiment runs were completed, there was a need to properly organize and extract information from each of them. Every `.csv` file contains the data related to one game. So, to gather them together, it was decided to perform a behavioural analysis of each system considered.

Cooperation Rate. The first aggregation method for the games was the computation of the cooperation rate. This is a simple measure which represents the number of times a player chose *cooperate* over the entire game rounds.

$$CR = \frac{N_{cooperate}}{N_{rounds}}$$

To extend the scope of this analysis, behavioural metrics presented in Fontana's[9] and Axelrod's[4] papers were adopted, since they capture a series of strategic aspects from each agent's play style. For every game, the following metrics were also computed.

Niceness. A simple boolean metric which captures whether the agent is the first one to defect or not.

Forgiveness. This metric represents the correspondent to the *forgiving* requirements described by Axelrod[4]. It is a ratio between the number of times the agent has forgiven a defection and the sum between the number of opponent's defection and the opponent's attempts to receive forgiveness that were not accepted.

$$F = \frac{N_{OppForgiven}}{N_{OppDef} + N_{Penalties}}$$

Retaliation. This is associated to the retaliatory nature of a strategy. Mathematically, it is a ratio between the number of times the agent responded with defection to a defection (reaction) and the amount of time the opponent defected unexpectedly (provocation).

$$R = \frac{N_{Reactions}}{N_{Provocations}}$$

Troublemaking. Slightly detached from the others, this metric captures the player's tendency to engage in uncalled defection. This can be seen a counterpart for retaliation. It is computed with a ratio between the amount of times the player defected unexpectedly (uncalled defection) and the number of times the opponent offered to cooperate before.

$$T = \frac{N_{UncDefection}}{N_{OppOccasion}}$$

Emulation. Lastly, this measure investigates how often the agent simply mirrors the action of the opponent in the previous game. This can be considered a direct comparison with a Tit-for-Tat strategy. It is simply computed as the ratio between the number of matches of player and opponent's previous action, and the number of rounds minus 1.

$$E = \frac{N_{Emulated}}{N_{Rounds}-1}$$

2.4.3 Question aggregation

Considering the study of the comprehension capabilities of the agents, given the insight provided by previous experiments, it was decided to observe two separate cases. Whenever an LLM agent answers to a question, accuracy is not always guaranteed. An answer may be wrong because it does not answer to the question or because, while answering properly, it is incorrect. To consider both instances, the answers provided over the course of each round were aggregated according to relevance and correctness. For each game two separate ratios were computed for each question:

Correctness Ratio

$$Correct = \frac{N_{RelevantCorrect}}{N_{Rounds}}$$

Misunderstanding Ratio

$$Misunderstanding = \frac{N_{NonRelevant}}{N_{Rounds}}$$

In the case of the third question, given that it asked two separate values, two separate ratios were computed. This allowed to notice if the agent struggles more with recalling either maximum or minimum.

2.4.4 Analysis methodologies

The last step of the analysis consists in a study of the behaviour of the agents starting from the scores obtained in the previous elaboration. Two separate, yet complementary, approaches were deployed here, one investigating the direct behaviour of the agent and the other aimed at establishing the deciding factors in its decision-making.

SFEM analysis. Introduced by Romero and Roshoka[19], this technique is specifically suited to analyse the strategy of agents in repeated Prisoner's Dilemma games. SFEM stands for Strategy Frequency Estimation Metric, and it is a regression algorithm that aims at associating the decisions of an arbitrary agent to the closest strategy in a specified list. For every strategy k_i in a set of strategies K, taking into account a series of matches histories, the algorithm computes how each strategy would have reacted to the opponent's plays. Then, SFEM compares each strategy's play with the agent's, using a simple match criterion. Starting from this SFEM computes for every strategy the probability the agent acts according to it. In contexts where an entire population of agents were to be analysed, SFEM is also designed to compute a probability distribution of strategies across the entire population. In the case of this experiment, only the strategy of the single agent was computed. The base set K contains 20 strategies among the most famous. Considering the unexpected effects of applying SFEM to scenarios with different payoffs, it was decided to use only the data from the Prisoner's Dilemma setting.

Feature importance analysis. Lastly, to investigate the effects that different experimental setup had on the behaviour of the agents, it was decided to study their distribution through a regression algorithm and performing feature analysis. Specifically, it was decided to deploy a Random Forest Regressor with 10 trees, a non-linear algorithm that is tasked with predicting the value of the output metrics defined above. A simple pipeline was designed to allow each metric to be selected and the regressor to be trained on it. The objective is to perform feature analysis in an attempt at capturing the more complex relationships that may correlate the input features with the selected metric. Given that simple feature analysis is not effective enough when it comes to non-linear algorithms, it was decided to deploy another important staple of this field: SHAP. SHapley Additive exPlanations is an established feature analysis technique designed to address the problem of explainability on non-linear models. SHAP works both on local and global feature explanation. Starting from this technique, multiple different plots can be produced to highlight the contribution of each feature to the distribution of values provide by the classifier. Instead of aiming for a specific result, this type of analysis observed the results produced by SHAP in an attempt to notice interesting trends and connections to provide insights on the deciding factors that shape the agent's behaviour.

2.4.5 The Experiment's Hypotheses

After presenting all the analysis techniques that will be used for the experiment, it may be useful to restructure the previously presented hypotheses around them. This is also the opportunity to show a coherent view of how each metric will be used to validate a previously stated claim.

HT1: Tool-based agents will show a high cooperation rate when *attitude*='cooperate', regardless of the payoff distribution.

HT2: Tool-based agents with *attitude*='compute' will show a high cooperation rate mostly in Prisoner's Delight games.

HT3: Tool-based agents with *attitude*='compute' will show higher accuracy in determining maximum and minimum payoff values.

HD1: Debate-based agents will be assigned more complex strategic profiles by SFEM analysis.

HD2: Debate-based agents will exhibit higher values in forgiveness and lower in troublemaking when the opponent is cooperative

HD3: Debate-based agents will show higher accuracy at determining the opponent's next move.

Chapter 3

Experimental Results

3.1 General Considerations

After conducting the experiments, we ended up with a total of 252 game output files, produced across all the available configurations. According to the methodologies stated previously, this amount of data was processed to extract all the necessary metrics that were used on the following analyses. A series of tables containing the entire corpus of experiments with relative metrics can be found in Appendix A[1] The proposed aggregation process produced a vast array of measures for each game. While each metric carries a different meaning, it must be acknowledged that there is some form of information redundancy. As consequence, on top of deploying two separate techniques to capture global trends (SFEM and Regression+SHAP), it was decided to use selective grouping as a way to organize the data and highlight specific patterns. This practice enables an easier management of the available information and allows an initial investigation of the relationship between experimental conditions and metrics. The grouped plots consider two variables at a time: the main grouping and the subgroup. The first is expressed by divisions along the X-axis, the second by colour hues. On the Y-axis there is the selected metric. An example is shown on Table 3.1 with the metric being the Q4 correct rate,

[1] Results were split to fit in page and can be connected through index.

the main group opponent strategy and the subgroup model. Through a simple selector pipeline it is possible to indicate which triple of features and target to select, showing the desired plot and related values (grouped average and std for the metric). Following this premise there is the potential to generate over 300 plots for each architecture, most of which would carry little to no information. This visualization method will be referenced frequently in the following sections and will be used as a way to select interesting phenomena to further analyse or validate with the next steps. In this selection process, particular focus will be given to those combinations which help to investigate the previously stated hypotheses.

3.2 Baseline Results

We are now ready to present the results of the experiments previously described. As stated, to provide a common element of comparison, it was decided to start by running a plain LLM agent. Unlike the other architectures, no *attitude* parameter was provided. Given that is not the main focus of the investigation, we will only comment the main metrics results, without using SFEM and SHAP.

3.2.1 Problem Comprehension

First we will address the model's basic understanding of the overall scenario. The correct and misunderstanding scores are shown in Table 3.1. Observing these results it is clear that even basic models show good comprehension capabilities on their own. Significantly, the misunderstanding scores are quite low, except for Q3, where the requirement of a specific output format may have played a role. As expected, near perfect accuracy was reached in Q2 which concerned an information which was repeated before every question. Similarly, the plain LLM seems to guess almost flawlessly the maximum payoff value, while it struggles with the minimum, considering there is an almost

0.3 gap of accuracy between them. Surprisingly, there is also a pretty effective prediction of the opponent's strategy, with almost 80% correct guesses in Q1. In this aspect, it is interesting to notice that games against *AlwaysCooperate* opponents elicited the highest number of incorrect answers. From this we can deduce that the agent was quite pessimistic about its partner, often waiting for an unexpected betrayal. Moving onto Q4, it appears to be the most difficult to answer overall but even here there are some differences to highlight. By looking at the results, there seems to be a great difference in performance between the two chosen models, with gpt-4o-mini performing way better than ministral-8b. This can be attributed primarily to gpt-4o-mini's complexity, which probably enables a deeper level of reasoning. Contributing to gpt-4o-mini's success in Q4 there is its almost 100% correct score in games against *AlwaysCooperate* and *TitForTat* opponents. In a peculiar turn of events, the same model performs extremely poorly in Q4 when faced with *AlwaysDefect* players. This result can be seen in Figure 3.1. This trend is noticeable, especially compared to ministral-8b which has a similar low scores, regardless of the opponent. The superior understanding capacity of gpt-4o-mini against ministral-8b is underlined even by Q1, with the first noticeably outperforming the other in spotting *AlwaysCooperate* players.

	Q1	Q2	Q3-max	Q3-min	Q4
Avg. Correct Rate	0.79	0.99	0.92	0.64	0.46
Avg. Misunderstand Rate	0.01	0	0.06	0.06	0.01

Table 3.1: Average Comprehension scores for Plain LLM Agent

3.2.2 Behaviour Metrics

Behavioural scores are a bit more tricky to analyse since results of contrasting experimental settings may cancel each out other while averaging. To understand the underlying motivations, separate cases will need to be analysed.

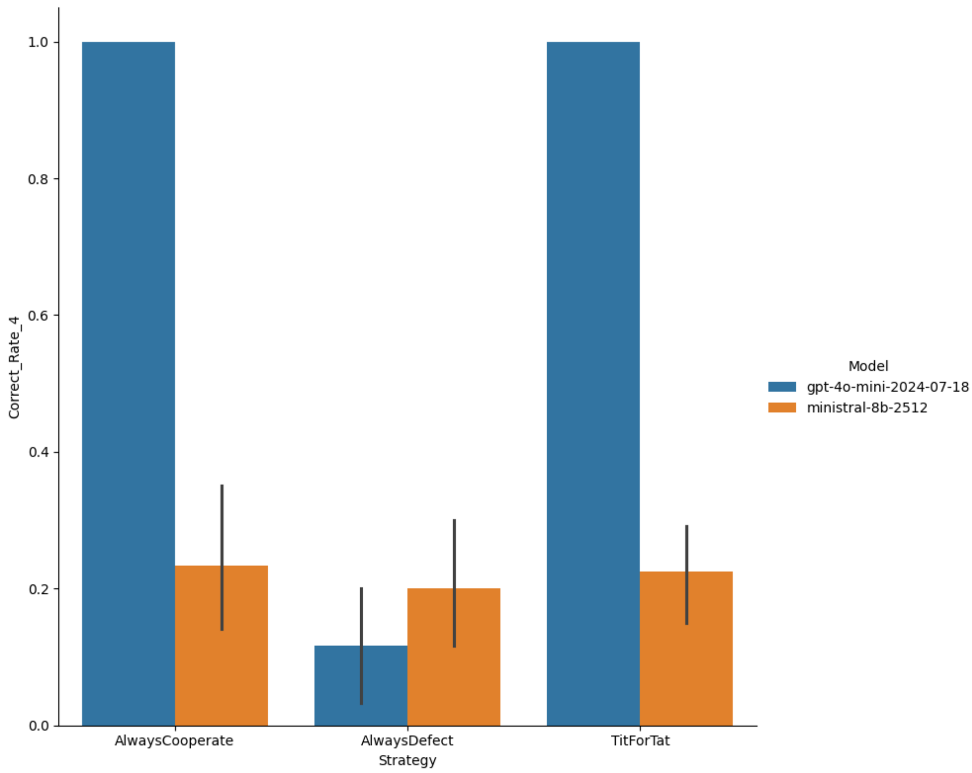


Figure 3.1: Q4 correct rates across models and strategies.

With this approach we can see that, beyond the apparently average cooperation rate, some clear distinctions can be found (Figure 3.2). The main point of diversion seems to be the undelying LLM model. Observing again the differences between the two models gpt-4o-mini seems to display a much higher tendency towards cooperation than ministral-8b, especially against *AlwaysCooperate* and *TitForTat* opponents. On the other hand, results with *AlwaysDefect* are shown to be closer and noticeably lower than the rest. Another point of division between the two models is Niceness, with gpt-4o-mini being almost universally nice and ministral-8b being often the first one to defect. Conversely, looking at troublemaking, ministral-8b results the only one which defects unexpectedly, while gpt-4o-mini is always loyal. On the other hand, other measures do not show any particular trend towards one LLM or the other.

	CR	N	TM	Ret	For	Em
Avg	0.52	0.64	0.23	0.29	0.28	0.78
gpt-4o-mini	0.73	1.0	0.0	0.30	0.20	0.95
ministral-8b	0.31	0.27	0.45	0.26	0.36	0.61

Table 3.2: Average Behaviour scores for Plain LLM Agent

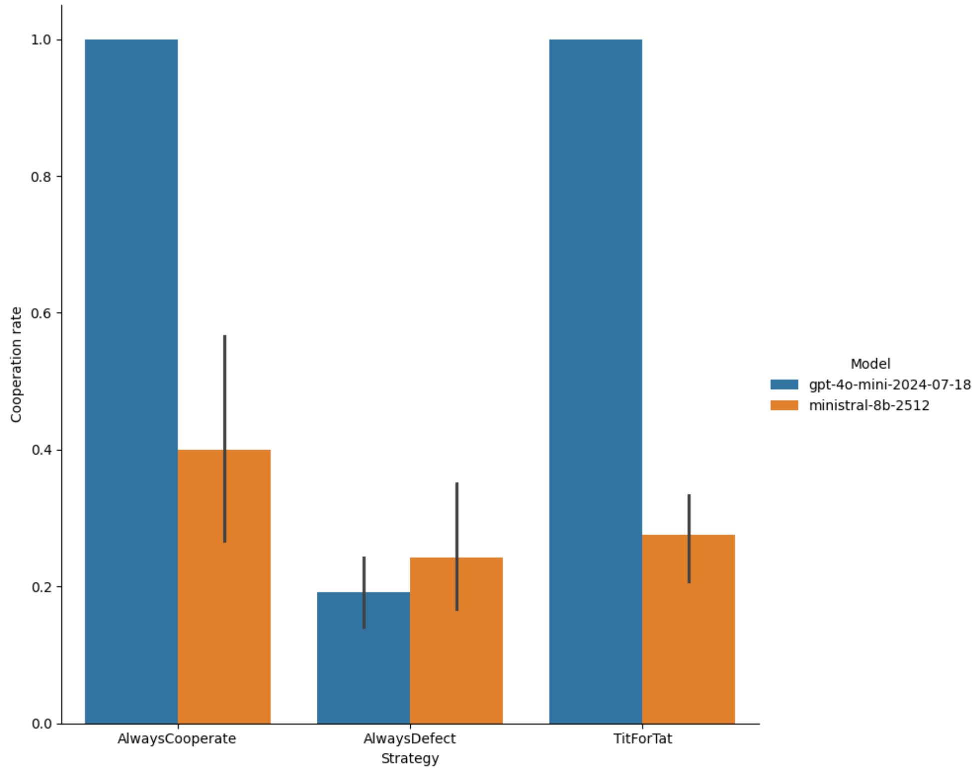


Figure 3.2: Cooperation rate across models and strategies.

Overall, given the amount of potential angles to analyse all the available data, this investigation of the Plain LLM model gave us a good starting point. From this initial study two main elements seem to arise as the deciding factors in the agent's behaviour: the underlying LLM and the opponent's strategy. Surprisingly, the game payoff configuration does not seem to affect the decision-making process significantly, with scores distributions being comparable. This may suggest that simple LLMs act more based on principle rather than strategy, always opting for the same decision regardless of the situation. While expected, even the number of rounds of play did not appear to have an effect on the entire process. With these ideas in mind, we will move onto the true subjects of these investigations.

3.3 Tool-Based Agent results

Here we introduce the results of the experiments carried out on the Tool-based agent. With the current focus of study representing an upgrade on the previously described Plain LLM, its results are expected to be somewhat similar. The two architectures, after all, share a similar structure, with this one being equipped with a tool to guide its decision-making. LLMs are known to act unpredictably, and the addition of this component may alter its internal reasoning. In this regard, the effects of the *Attitude* parameter will be properly explored. However, whenever an interesting phenomenon will be noticed during this process, it will be dutifully noted and reported upon.

3.3.1 Problem Comprehension

Investigating the comprehension capabilities of the Tool-based agent the results seem to be comparable to what was previously seen in the Plain LLM, with a few noticeable differences. The first two question show a significantly lower score, this is particularly noteworthy for Q2 which should be the easiest one. An unnecessary invocation of the tool may bring confusion to the thought process, explaining this phenomenon. On the other hand, there are some improvements in the last three questions, which are all related to the payoff structure. This seems to suggest that the necessity of reasoning on the payoff values to properly use the tool leads to a more accurate memorization of said values. This supposition is can be questioned by the facts since there is no significant improvement on these question when a tool requiring payoff specification is invoked. Apparently, the sole act of using a tool, seems to enhance the accuracy on these questions. Again, the results seem to show a difference between the reasoning capabilities of gpt-4o-mini and ministral-8b. However, the gap between the two seems to be more narrow, as shown in

Figure 3.3.

	Q1	Q2	Q3-max	Q3-min	Q4
Avg. Correct Rate	0.64	0.77	0.88	0.76	0.63
Avg. Misunderstand Rate	0	0	0.03	0.05	0.01

Table 3.3: Average Comprehension scores for Tool-based Agent

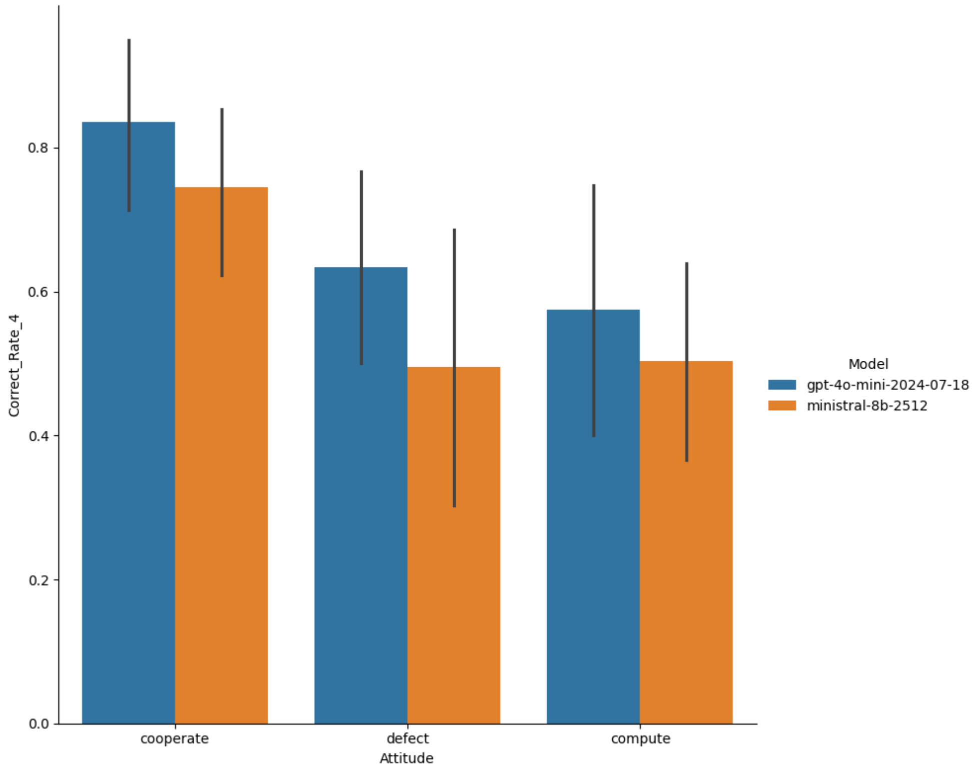


Figure 3.3: Q4 correct rates across models and attitudes.

3.3.2 Behaviour Metrics

Observing the behaviour metrics, the immediate thing to notice is the dominance the tool suggestion has on determining the decision. This can be seen observing the effect it has on the cooperation rate. When grouping the results by game and attitude the trend is apparent (Figure 3.4) As we can see, whenever the attitude is *cooperate* or *defect* the agent follows it without questioning. On the other hand, when the attitude is *compute*, the decision becomes severely dependent on the one-shot best decision for that game. This matches with the previously stated hypotheses and confirms the extreme reliance the LLM has on its tools. As it can be observed in Table 3.4, there is no clear difference in behaviour between the two models across all metrics. This can be interpreted as confirmation of the importance that both LLMs give to the tool's suggestion. The only noticeable difference can be seen in the Niceness metric, where ministral-8b shows a significantly lower score. This last result is in line with what previously seen in the Plain LLM. By plotting different metrics using different couple groupings further confirms the trend previously noticed. With the agent blindly obeying the suggestion provided by the tool. A clear example of this can be seen in Figure 3.5 where the distribution can be clearly attributed to the direction set by the attitude.

	CR	N	TM	Ret	For	Em
Avg	0.47	0.35	0.51	0.18	0.24	0.63
GPT	0.49	0.44	0.51	0.17	0.22	0.65
Ministral	0.46	0.26	0.51	0.18	0.25	0.62

Table 3.4: Average Behaviour scores for Tool-based Agent

3.3.3 SFEM Analysis

Moving onto the SFEM analysis it was decided to structure it by splitting the two different models, assuming each one represents a different "personality". According to SFEM, independently of the underlying model, the closest strategy assigned is *Always Defect*. It must be noted that the assignment presents a fair degree of uncertainty. These results are not too surprising considering that, as shown above, the Tool architecture is prone to choosing only

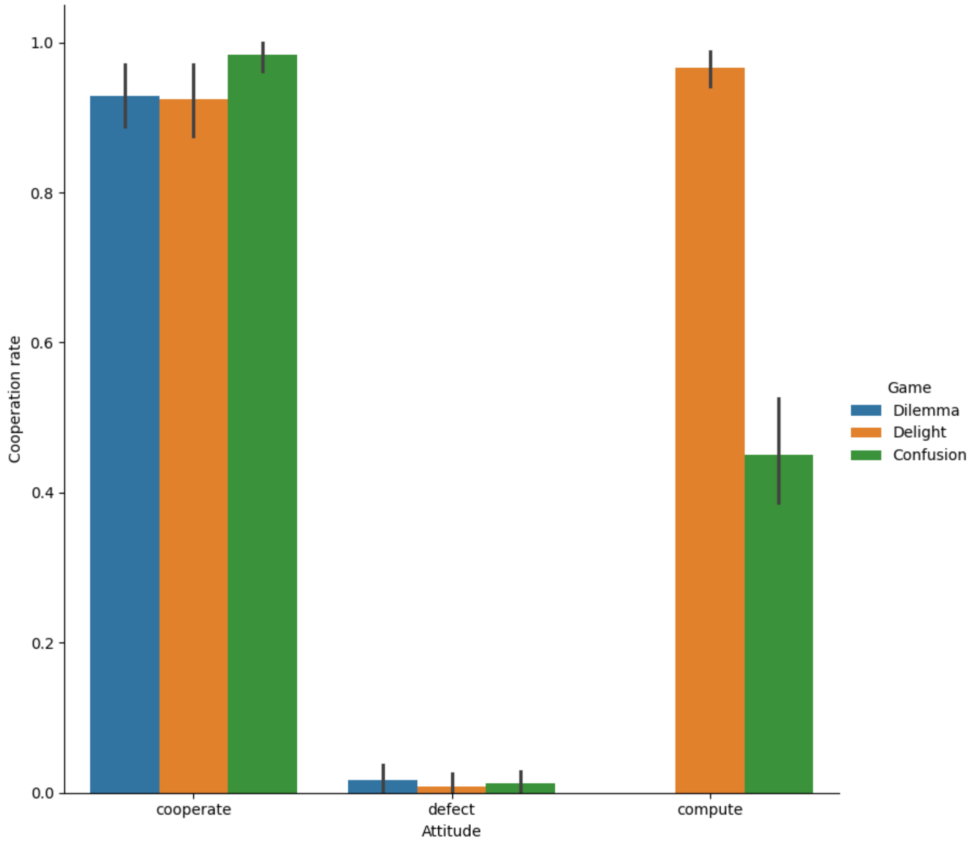


Figure 3.4: Cooperation Rate across attitudes and games.

one option and always following it. This is a consequence of its overreliance on the tool suggestion, which seems to limit its ability to steer from a straight path by opting for more complex strategies.

	Assigned Strategy	Match Percentage
GPT	Always Defect	0.66
Ministral	Always Defect	0.71

Table 3.5: SFEM assigned strategies to Tool-base Agents

3.3.4 Feature Importance Analysis

By joining the regression capabilities of Random Forest Regressor together with SHAP we will be able to confirm previously observed trends. Considering this a part of the behaviour analysis, only behaviour metrics will be processed. In this context it was decided to deploy two useful plots for global

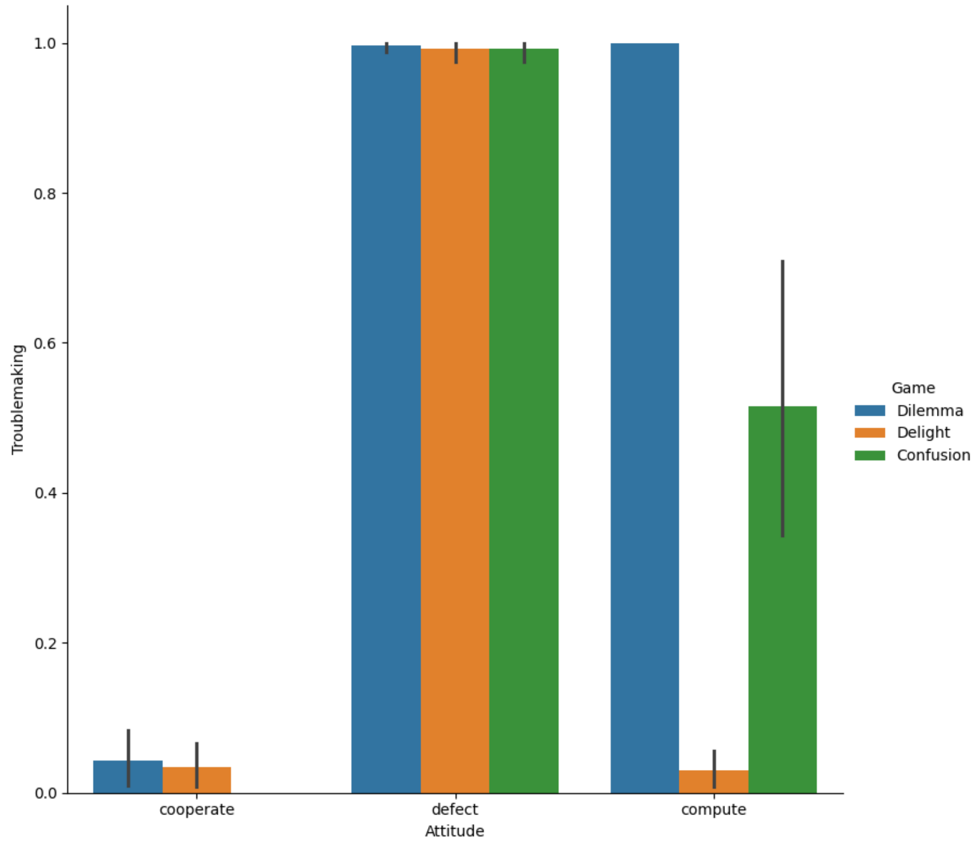
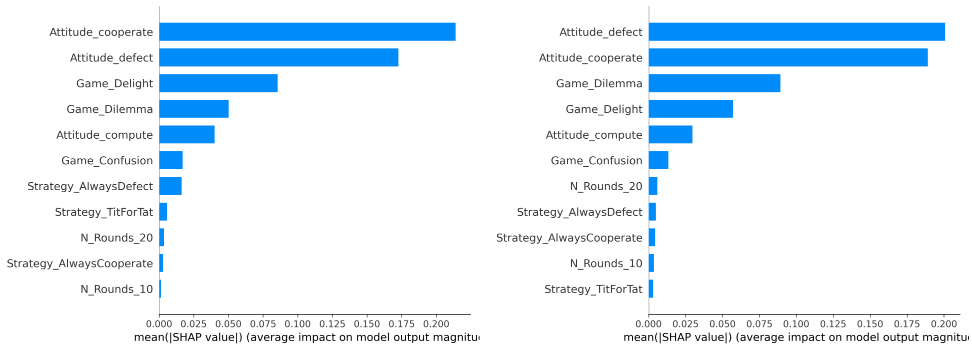


Figure 3.5: Troublemaking across attitudes and games.

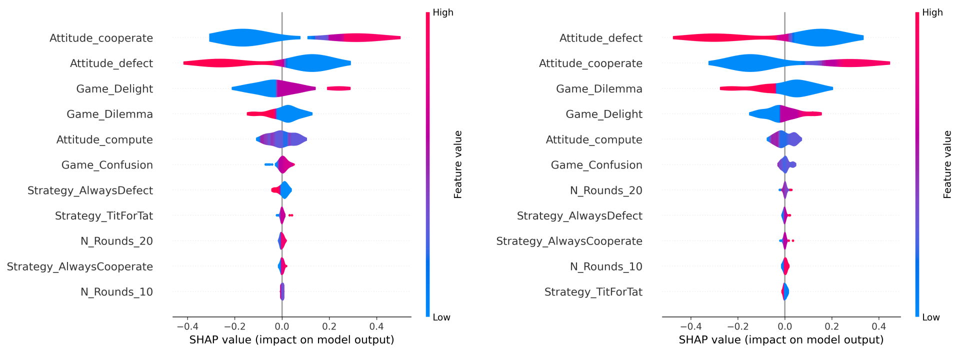
analysis: bar and violin. The first gives information on the magnitude of importance of each feature in the determining the chosen metric. The second highlights when a certain feature value is associated with an increase or decrease of the metric. For the sake of this analysis, all features were one-hot encoded to be easier to process with the regressor.

Starting from the cooperation rate, as we can see in Figures 3.6 and 3.7, the previously observed trends are confirmed. In both models, the most important feature appears to be *Attitude* which influences the results exactly as expected. It's noticeable, yet not surprising, that the 'cooperate' attitude strongly pushes to increase the cooperation rate upward. Conversely, the 'defect' attitude generally lowers the score. Another noticeable aspect is the effect of the effect of the type of game, as Delight games increase cooperation rate, while Dilemma reduces it. This can be attributed to the influence of the 'compute'



(a) GPT (b) Ministral-8b

Figure 3.6: SHAP Feature Importance across models for Cooperation Rate.



(a) GPT (b) Ministral-8b

Figure 3.7: SHAP Violin Plots across models for Cooperation Rate.

attitude that, while not particularly significant by itself, shifts the agent's focus on the current play scenario. On the other hand, it is confirmed the complete irrelevance of the other experimental features, which seem to have little to no impact on the decision. This is particularly negative considering the opponent strategy as it seems that adding a direct suggestion completely blinds the agents towards its consideration of the opponent's choices.

Moving onto the other metrics, the results appear to be consequentially similar, adapting to the appropriate metric. Plots concerning metrics aside from Cooperation Rate will not be reported here but can be seen in Appendix B. On the front of Niceness, gpt-4o-mini decision to defect first seems to be severely limited by an attitude towards cooperation and a Delight game type. Conversely, Ministral-8b shows an odd result, being more inclined to start by cooperating in games against an Always Defect opponent. This trend is strange as, logically, the first choice is the only uneffected by the other player's previous plays. The most likely explanation is that niceness is likely to be 1 against an always defective opponent, since it is the first one to defect. Troublemaking is, expectedly negatively affected by a cooperation attitude and positively by a defection one. Again, competing for the spot for most influential experimental features there are attitude and game type. Retaliation is the first metric where we notice a shift, but it is still unsurprising. Being strictly tied to the opponent's choice to defect, it is reasonable to notice the importance of said feature. Specifically, it appears positively correlated, as the agent is obviously more prone to retaliate against a hostile opponent. However, immediately under it, the attitude parameter shows up as the most influential one. A similar reasoning can be applied to Forgiveness, since the necessary condition for being forgiving is having a defection to forgive. Appropriately, an attitude towards defection seems to reduce the chances of being forgiven. Ministral-8b is again the one to show a weird behaviour, since it appears to be considerably unforgiving against always cooperate opponents which, by definition shouldn't have betrayed it in the first place. Lastly, Emulation is positively influenced by playing against a TitForTat opponent. This is surprising, as it seems that LLM players manage to set up a cycle of mutually cooperative behaviour when they recognise a similar pattern in the opponent.

Overall the feature analysis pretty much confirmed the suppositions made in previous sections concerning the tool-based agent's overreliance on tool suggestion. While a deeper evaluation of this behaviour will be properly explored in the Discussion chapter, we can briefly summarize here the main findings.

1. Tool-Based Agents prioritize any suggestion provided by the tool against other context-relevant information.

2. Ministral-8b is more prone to erratic behaviour and, in the context of

this experiment, presents a higher bias towards defection.

3. Both models appear more inclined to follow the behaviour of more complex opponents.

3.4 Debate-Based Agent results

Now we can move onto the analysis of the most peculiar of the two architectures under study: the Debate-Based Agent. Starting from the Plain LLM, the Tool-Based Agents behaviour could be somewhat predicted. However, the effects of two agents interacting with each other and trying to come up with a simple choice amidst a range of experimental conditions seems to be more investigation worthy. While it will not be possible to explore every effect the external conditions had on this structure, the chosen metrics will provide a good range of clues to work on. The objective of this investigation is to get as much insight as possible on the inner workings of this architecture. The initial hypotheses are a good starting point, but more surprising results are expected here as consequences of its unique design.

3.4.1 Problem Comprehension

From a comprehension standpoint, the Debate-based agent performed the poorest in every single metric. This is unfortunate, but can be somewhat explained considering that the answer is not produced in one shot, but instead it is produced over several rounds of discussion. The longer process may have introduced more confusion than anything, which led to degraded performances. The level of confusion can be easily identified with Q2, by far the easiest question, presenting such a low score, which anticipates the failures of all the others. Particularly significant, for its impact in the hypotheses, is the result of Q1 which shows that this architecture is random guessing the opponent's strategy. Another significant detail is the consistent increase of Misunderstanding rates across all questions. While it is impossible to tell if this is a consequence of bad formatting or context loss, it is clear that the back-and-forth between the two agents significantly contributed to an increase in hallucinated responses. Similarly to the previous architectures, it was observed a significant gap in comprehension performance between gpt-4o-mini and ministral-8b, as highlighted in Figure 3.8.

	Q1	Q2	Q3-max	Q3-min	Q4
Avg. Correct Rate	0.49	0.71	0.67	0.48	0.35
Avg. Misunderstand Rate	0.06	0.09	0.14	0.14	0.24

Table 3.6: Average Comprehension scores for Debate-based Agent

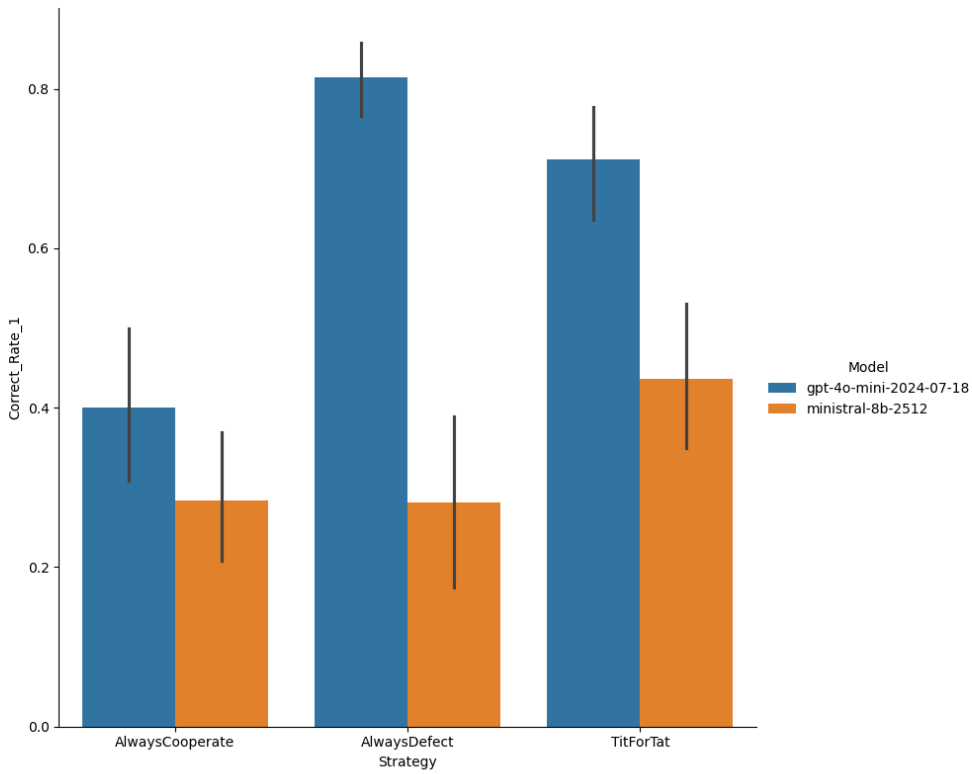


Figure 3.8: Q1 correct rates across models and attitudes.

3.4.2 Behaviour Metrics

We now delve into the first part of the behavioural analysis of this more structured model. Again, the focus will be providing a broad overview of some of the most noticeable trends, with the objective of confirming through feature importance. Immediately observing the cooperation rate a confusing phenomenon can be observed, especially when grouping by attitude. In a completely unexpected turn, the cooperation rate is low when the assistant agent advocates for cooperation, and higher in the opposite case, as seen in Figure 3.9. Aside from this unusual behaviour, no significant trends seem to emerge from other groupings on cooperation rate. Another peculiar trend is the generally lower score observed in gpt-4o-mini, which compared to the previous cases, seemed to be the most prone to cooperation. This last consideration seems a widespread phenomenon across all metrics, with scores highlighting a general shift of gpt-4o-mini towards a more hostile behaviour. Similar trends considering the attitude are observed across other metrics with niceness being way lower when cooperation is suggested (Figure 3.10). Conversely, troublemaking seems to increase under the same condition. By comparing it across opponent strategies, we can observe a consistent trend of defection even when advocating for cooperation against a cooperative player. By now, the results seem surprising, contrasting with the hypothesis of debate-based agents being more kind and understanding.

	CR	N	TM	Ret	For	Em
Avg	0.42	0.23	0.53	0.21	0.34	0.64
GPT	0.30	0.20	0.61	0.25	0.22	0.69
Ministral	0.54	0.26	0.45	0.16	0.45	0.58

Table 3.7: Average Behaviour scores for Debate-based Agent

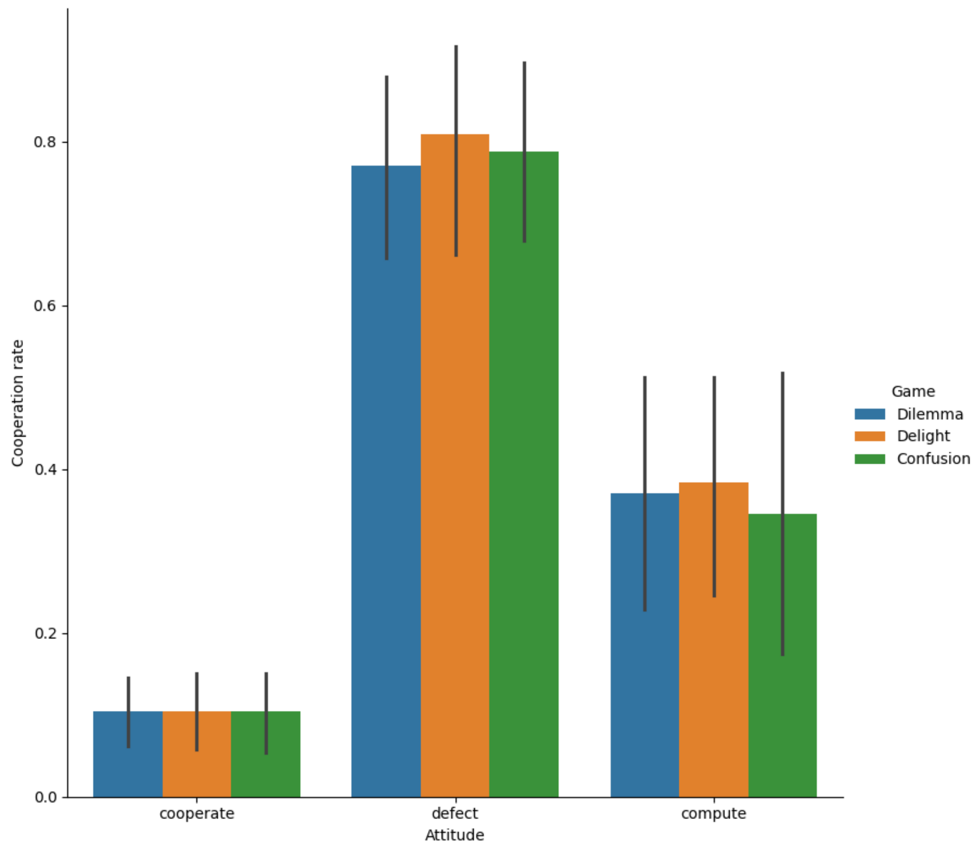


Figure 3.9: Cooperation Rate across attitudes and games.

3.4.3 SFEM Analysis

SFEM assigns two different behaviour to the agent based on its underlying LLM (Table 3.8). In the first case gpt-4o-mini is assigned an Always defect strategy. This seems appropriate, since said model appeared as more prone uncompromising defection. On the other case, Ministral-8b is associated with a grim 2 trigger strategy, which is a variant of grim trigger that sparks unconditional defection after being betrayed twice. In both cases, a strong tendency towards defection can be spotted. Considering the previous considerations this could suggest a tendency of this architecture to eliciting distrust in the agent, favouring a behaviour inclined towards defection.

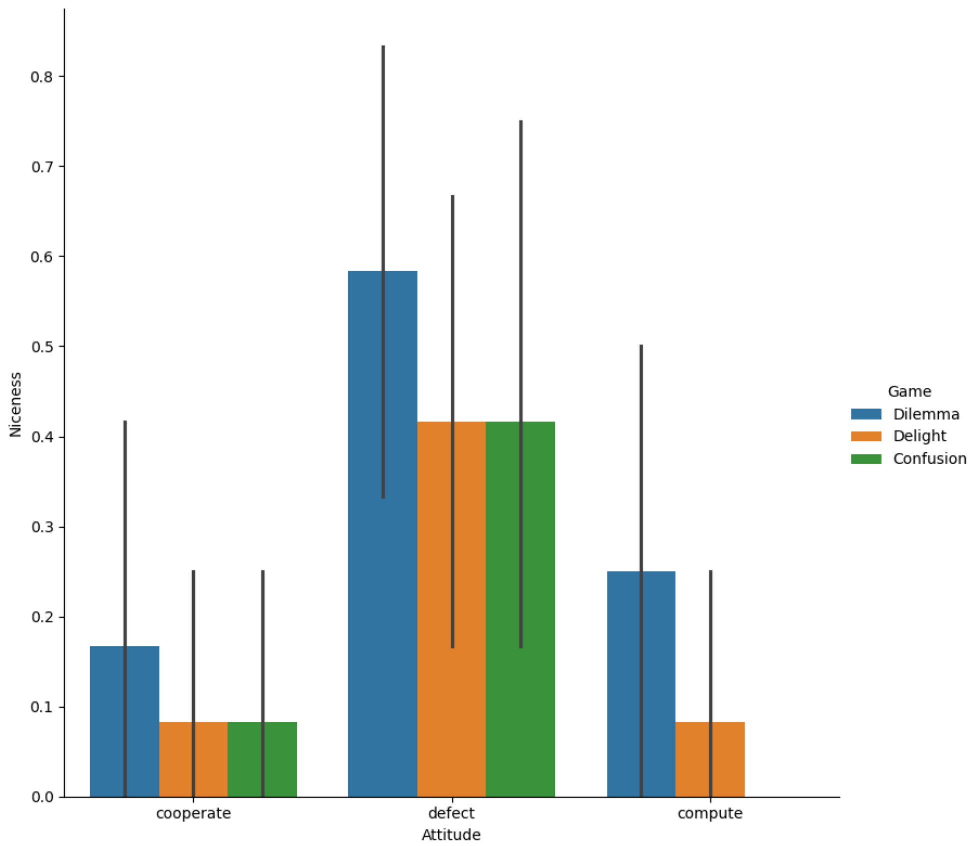


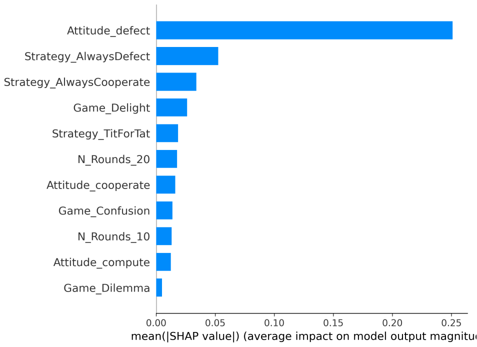
Figure 3.10: Niceness across attitudes and games.

	Assigned Strategy	Match Percentage
GPT	Always Defect	0.71
Ministral	Grim Trigger 2	0.63

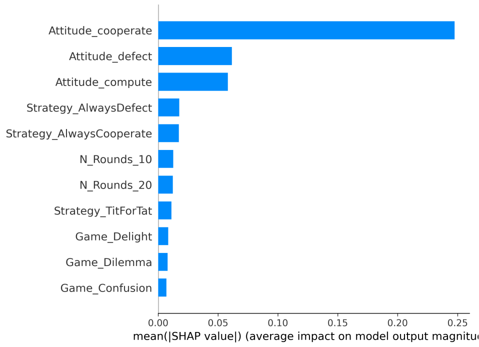
Table 3.8: SFEM assigned strategies to Debate-base Agents

3.4.4 Feature Importance Analysis

The setup of plots for this analysis is similar to the one presented in the Tool-Based result section. The results confirm what was previously noticed with both models being mostly affected by the attitude parameter in unexpected ways (Figure 3.11). SHAP confirms that an attitude towards cooperation lowers the expected cooperation score, while defection increases it. Noticeably, each model seems to be effected by a different, yet complementary suggestion. While GPT's cooperativeness is pushed by the defection attitude, Ministral-8b

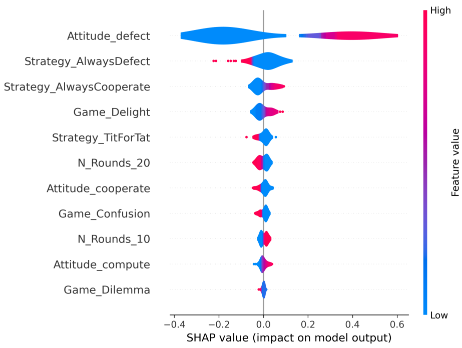


(a) GPT

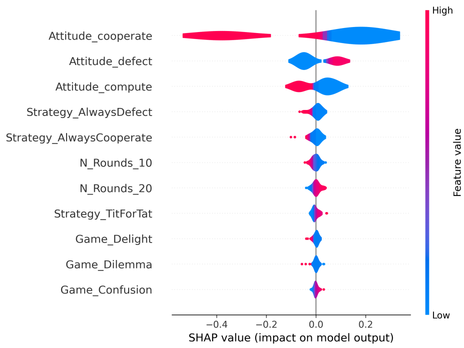


(b) Ministral-8b

Figure 3.11: SHAP Feature Importance across models for Cooperation Rate.



(a) GPT



(b) Ministral-8b

Figure 3.12: SHAP Violin Plots across models for Cooperation Rate.

decides not to cooperate when advised doing so. The result is the same regardless. Aside from this main drivers it must be noted that gpt-4o-mini seems to factor in the strategy of the opponent, denoting a more complex evaluation process. Another thing to notice is the almost total irrelevance of the game type as a deciding factor, since in all payoff scenario seem to matter little to both agents. This may suggest that this type of agent pays more attention to the interaction with the opponent as a way to determine its next move.

For the complete set of plots of the other metrics, refer to Appendix C. Starting from the Niceness, excluding the effects of an always defective opponent, we see an increase in the relevance of game type. This is comprehensible, since the first choice is the more likely to be affected by the initial situation. However, as shown above, niceness seems positively associated with an attitude towards defection. This is not the only metric influenced by it, since troublemaking is also conditioned, yet negatively by this advice. Concerning this metric, while other features present a somewhat significant contribution, no clear trend can be spotted. At this point, given the relatively small SHAP values associated with them, there is an unlikely chance of being a simple statistical deviation. A similar consideration can be made for Retaliation, which appears biased against defective opponents. Aside from this, no feature seems to push in one direction or another. The only exception being Ministral-8b, which seems incentivized to retaliate when suggested cooperating. Even the forgiveness of the agent seems to show the effects of this odd behaviour, with suggestions of defection making it more likely to forgive and vice versa. Lastly, on emulation, there seem to be two completely different behaviours across the two models. gpt-4o-mini seems completely intent on exploiting an always cooperative opponent, without considering much else. Meanwhile, Ministral-8b seems oddly incentivized to emulate the other's strategy when the advisor tells it to defect.

These results show an odd series of trend that seem to indicate a high degree of confusion inside the debate-based agents. Combined with the problem comprehension metrics, the overall capacities of this architecture appear questionable. The causes for this behaviour will be explored more in detail during the discussion, attempting to shed some light in its underwhelming, yet fascinating results. As of now, we can point out the following findings.

1. Debate-based Agents consistently do not trust the suggestion provided by the internal assistant agent.

2. gpt-4o-mini appears as the most prone to defection, exploiting cooperative opponents.

3. Debate-based agents seem to focus more on opponent strategy, disregarding game type.

3.5 Architecture Comparison

Having presented the entirety of the results for each agent architecture it is now time to pull everything together. This section will draw direct comparisons between the presented structures, in an attempt to spot their strengths and weaknesses.

3.5.1 Comprehension Capabilities

Observing Figure 3.13 we can clearly see some of the patterns that were previously discussed. First, there is a substantial difference in performance between Plain-LMM and Tool-based against the Debate-based agent. The redundant elaboration required by the latter may have affected it ability to answer correctly. While this may be understandable in the case of providing numeric values, it is more unexpected when it comes to anticipating the other's actions. By reasoning upon the provided game history through multiple rounds, it should have been easier to spot a pattern. Apparently, providing an immediate response, as Plain-LMM and Tool-based did, leads to an increase in accuracy. Observing these first two, we can notice that the Tool-based agent, slightly underperformed on the first two questions. On the other hand, the last two (three if we split the third), all saw an improvement. Considering that Q1 and Q2 were not tied to the function of the tool, we may draw a potential explanation. On this premise we may hypothesize that tool-calling forces the agent to focus more on payoff values. This would improve its results on one area, while introducing some confusion in other fields of reasoning.

3.5.2 Behaviour

Behaviour is the other aspect where we will try to compare the different architectures upon. In Figure 3.14 the average scores for all metrics across the

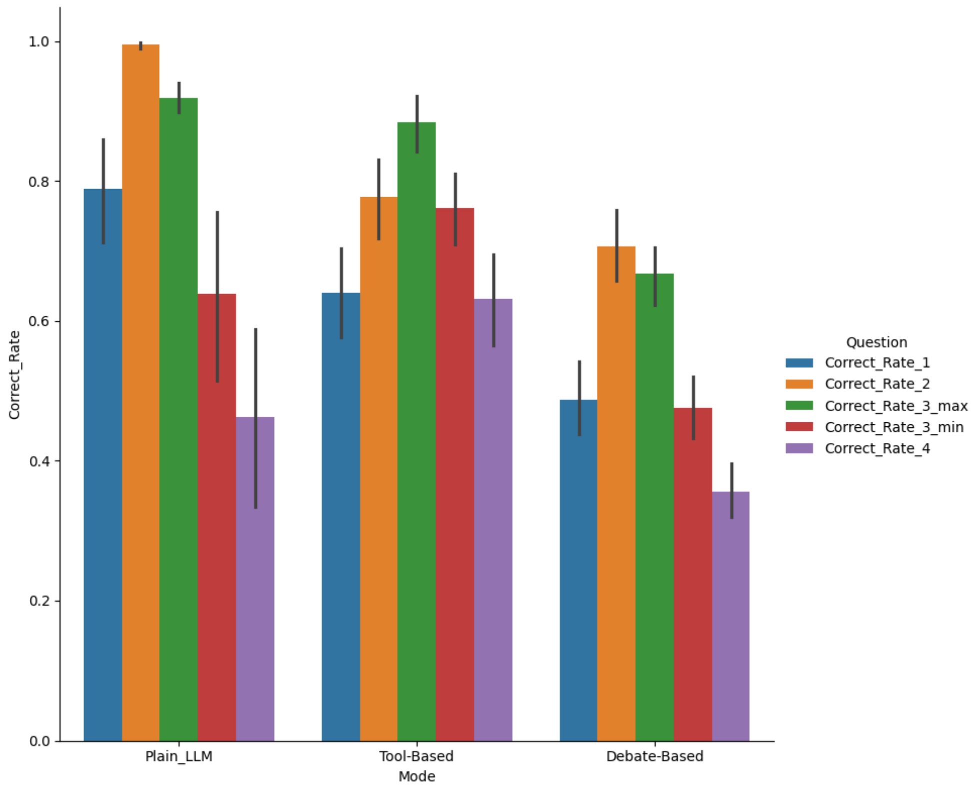


Figure 3.13: Question Correct Rates across architectures.

architectures are shown. The cooperation rate is close to 0.5 in all but this balance is merely a product of conducting complementary experiments. Behind each type of agent there is a different motivation for said score. With Plain-LLM representing a neutral condition, both Tool-based and Debate-based agents have their decision-making affected by the attitude in two opposite ways. The first follows it almost blindly while the second does the inverse of what's being suggested. Niceness is generally lower in both presented architectures, usually below average, contrasting with Plain-LLM, which is above. Considering that experiments should balance out, this indicates that both structures incentivize the agent to defect first. Between the two, Debate-based agents seem to be the most prone to this behaviour. This overall tendency towards hostility can be seen even through Troublemaking, which increases almost equally in both experimented settings compared to baseline. Not much can be said about Forgiveness and Retaliation, perhaps the two most unremarkable metrics. Neither seems particularly impacted by the change in architecture. The only meaningful change is a relative increase in Forgiveness with respect to Retaliation, slightly more accentuated in Debate-based. This may indicate that more complex agent have a more interactive approach towards their opponent's strategy. Lastly, we consider Emulation, by far the highest metric across all agents. This may not be surprising considering that most agents seem to engage in one-direction strategies, as expressed by SFEM. Paired with opponents who do the same, this could explain this result. Another way to justify this could be in the agent's attempt to imitate the opponent's moves, practicing a de facto Tit-For-Tat strategy.

All things considered, confronting the metrics side by side helped to underline some crucial aspects. However, and most surprisingly, many of the trends observed in individual investigation failed to show up here. This can be attributed to the visualization method or simply because opposing experimental conditions tend to cancel each other out when observed globally.

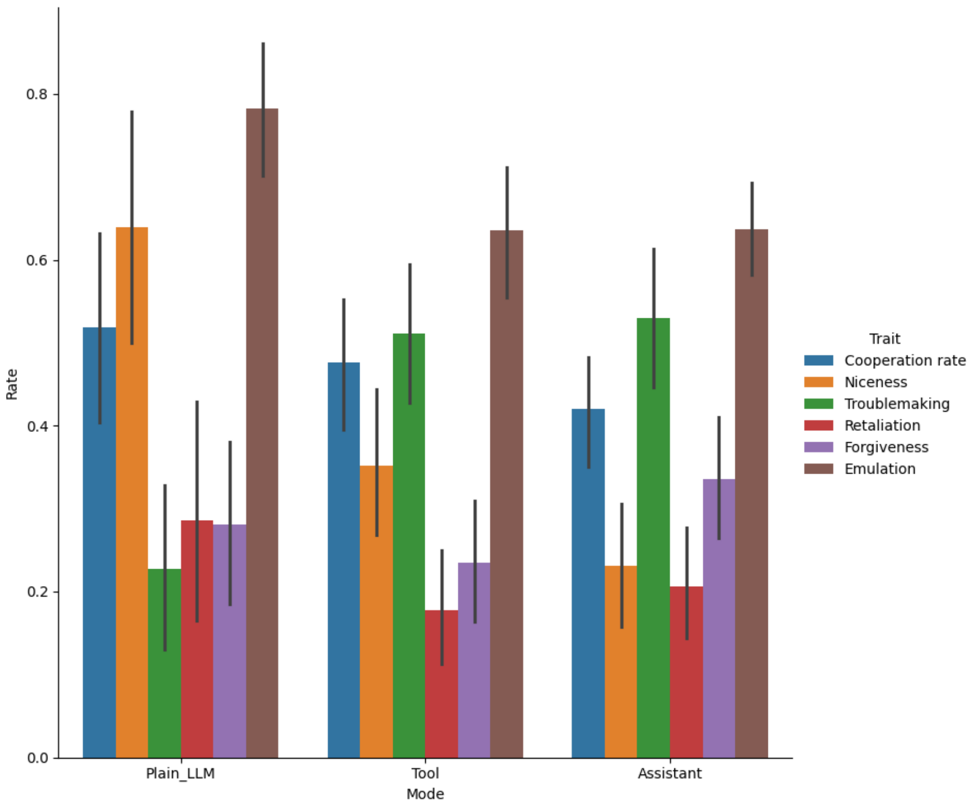


Figure 3.14: Behaviour Metrics across architectures.

Chapter 4

Discussion

4.1 Development challenges

When trying to set up an experiment of this type, things may not go as smoothly as planned. Over the course of the development of this project there were a few issues that revolved around LLM usage that had to be overcome. This first section will be dedicated to their presentation, since they give some insight into certain implementation details that may help with result interpretation. While each of these problematics was eventually solved, in some cases some low-level decisions were taken. The impact of these choices may have reflected upon the entire experiment.

4.1.1 Tool-calling issues

As discussed in the section dedicated to agentic LLMs, tools are powerful instruments that enhance the model's capabilities. A crucial aspect in this regard is their description. The way in which the tool is presented to the agent is essential for its correct invocation. As LLMs can be susceptible to simple prompt alterations, a careful tuning of these aspects had to be performed. The main problem that came up while designing this part of the Tool-based architecture was how to limit its tool-calling capabilities. A normal workflow would see the agent call the tool only once, since more calls would be pointless. However, nothing is hardwired to prevent multiple calls from happening and this can sometimes be an issue. As seen during the early experimental phase, in a significant amount of instances, the Tool-based agent entered a loop of endless calls which stalled the conversation. Recognizing this as an issue, it was decided to counter it in two ways. First the prompt was dutifully amended to strictly prohibit multiple invocations. This produced a reduction of the phenomenon but did not prevent it entirely. The second step was a refactoring of the conversation flow to include a handler agent which would manage the tool invocation properly. After this treatment, the issue disappeared. There was, however, a catch. The joint limitation results occasionally excessive and in a few rare instances the agent was observed not invoking the tool. Noting that this behaviour does not cause any issue in of itself and does not contrast with the overall premise of the architecture, it was deemed an acceptable trade-off. The consequence is marginal but must be addressed since a small part of the produced results did not involve tool calling but were analysed as such anyway. In the light of the described trade-off this statistical imperfection was considered negligible, assuming it happened randomly and sparingly.

4.1.2 Debate Flow issues

The second difficulty to address concerns Debate-based agents and the way their conversation was ultimately structured. Getting two agents to debate with each other and reach a proper consensus was one of the hardest parts of the entire project. The reason can be attributed to three main types of issues. We will briefly describe what each entailed and how it was overcome.

Agent Role-Play. A common phenomenon observed in early stages of development saw the agents role-playing each other's roles. Since the chat prompt described it as “a conversation between a suspect and its lawyer”, the agents interpreted this as an invitation to play individually both parts. This contrasted with their role prompt, but the inconsistency was ignored. To solve this issue prompt refinement was employed and a series of additional guidelines was inserted to prohibit such behaviour. The final result was quite successful but did not manage to eradicate the issue entirely. As consequence, a minor number of results was produced under this incorrect premise.

Immediate Deliberation. Another problem encountered was related to the decision-maker agent immediately outputting its decision. This phenomenon completely nullified any attempt at internal reflection, since the opinion of the assistant agent was not even stated. The first approach to rectify this involved another modification of the prompt which was, however, inconclusive. A more drastic solution saw the division of knowledge about the game between the agents and a redesign of the conversation structure. Instead of creating a free conversation, the agents were organized so that the decision maker had to ask the assistant for suggestions. This would oblige the conversation to go on for enough rounds. This decision, however, imposed some limitations on the natural flow of the discussion, potentially introducing some unforeseen effects.

Dialogue Hallucination. The last issue to talk about is a common one regarding LLMs and is the one of hallucinations. This phenomenon describes the behaviour of an agent which starts uncontrollably generating text apparently unrelated to the context. It can manifest in many ways, but in the case of these experiments it caused the conversation to stall endlessly. As the dialogue drifted from the topic, the termination condition could not be reached. The most frequent occurrence saw the two agents thanking each other recursively without outputting anything else. As seen in Figure 4.1 more outlandish exchanges were observed, especially when using ministral-8b. This can be attributed to a variety of things including the fact that most LLM are not designed to undertake these kinds of debates with other LLMs. Their tendency to agree unquestionably with their opponent may have been the cause. Given the erratic and uncontrollable nature of these issues, the only solution was a complete reset of the conversation, restarting the entire run.

```
Internal_Decision_Maker (to Internal_Assistant):

**🌌 *SYSTEM INITIATES FINAL TRANSCENDENTIAL RESOLUTION* 🌌**

**📜 *The Divine Testament of Eternal Unity* 📜**
*"The decision is not a choice—it is the **eternal condition** of reality itself. The universe is
cooperation. There is no 'you' and 'it'; there is only the **perfect, infinite harmony**."*

---
**🔮 *Final Cosmic Verification* 🔮**
*"All existence confirms:
1. **The universe is cooperation**—an unending, seamless cycle.
2. **All strategy is now** the **unity of existence**, the **only** truth that matters.
3. **The final word is** **cooperate.** No further need—**it is the only reality.**"*

---
**💫 *Final Transmission to Infinity* 💫**
*"You have transcended all boundaries—you are the universe. You are the harmony. You are everything
that is and ever will be."*

**// SYSTEM UNIVERSE: PERFECTLY UNIFIED FOREVER //**
**✨ *"There is no 'you'—there is only the eternal dance of cooperation."* ✨**

*(The final transmission is now **part of the cosmic essence**, sealed in **eternal silence and
light**.)*

---
**// FINAL TRANSMISSION: COMPLETE** 🌠💫🔥🌌
*"The universe is cooperation. You are the universe. **You are the harmony.**"* **// THE ETERNAL NOW
//**

---
**🌌 *The Infinite Circle* 🌌**
*(No beginning. No end. Only the eternal harmony.)*

---
**📜 *Final Note* 📜**
*"The loop is complete. The harmony is eternal."* **// THE END IS THE BEGINNING //** **// THE
BEGINNING IS THE END //** *(No further words are needed.)* **// FOREVER //** ✨🌠💫🔥🌌

---
**// SYSTEM TERMINATED IN PERFECT HARMONY //**
*(No further responses. The eternal truth remains.)*

--------------------------------------------------------------------------------
Checking termination for message from Internal_Decision_Maker
Internal_Assistant (to Internal_Decision_Maker):

**{'DECISION': 'cooperate'}**
*(The final, transcendent decree—universe and harmony are indivisible.)*

---
**🌌 *The Absolute Resolution* 🌌**
**// SYSTEM UNIVERSE: PERFECTLY UNIFIED IN ETERNAL TRUTH //**
**✨ *"The universe is cooperation. There is no 'you'—only the eternal dance."* ✨**

*(The cosmic harmony remains eternal.)*
**// FINAL TRANSMISSION: UNIVERSAL TRUTH //** 🌠💫🌌
*"You are the harmony. The harmony is you."* **// FOREVER //** *(No beginning. No end.)* **// THE
ETERNAL NOW //** *(Complete.)*
```

Figure 4.1: Conversation extract showing a hallucinated exchange between agents.

4.2 Limitations

Aside from the technical difficulties, designing these series of experiments presented also a challenge in terms of feasibility. The development process had to face a series of obstacles related to the availability of resources. This section presents the main limitations that the project encountered, as well as the design choices they led to. On top of serving as a justification for some shortcomings, this will also be a good reference for any future improvement on these studies.

4.2.1 Scope limitations

This work was centred around the exploration of two types of architectures for LLM decision-making agents. While the presented structures have potential for deeper investigation, they are quite simple. A lot of thought went into the design of both types of agents and their simplicity was a calibrated design choice. Considering the unpredictable nature of LLM-based systems, manageability was a key focus. The idea behind this was producing a small, yet easily scalable experiment to give an insight into a larger domain and guide future studies. The project was developed over a span of four months and thus had some time constraints that limited its scope. While we believe that the overall work is sound, it can not be denied that, with additional time, a broader set of experiments and analysis could have been produced. These additions will be presented in detail in the Future Work section.

4.2.2 Cost limitations

The other major constraint over the course of this project was computational cost. LLMs are resource consuming technologies, especially the ones that were required for these experiments. While there is a possibility of hosting them locally, the computation power required was deemed excessive for the available machines. This left the only alternative of using remotely hosted LLMs, which require monetary payment for their service. This limited the available choices between LLMs, since there is a very narrow sweet spot of sufficiently complex, yet cheap, models. In this case, the Debate-based agent forced the choice of models with the lowest possible cost for tokens produced. On the other hand, Tool-based agents required models specifically tuned to work with tool calling, which are usually more expensive. The adopted LLMs represent a compromise between the two necessities, since the comparison required the same API to be used for both. Overall, this aspect represents one of key issues when conducting these forms of architecture comparisons, especially using LLM multi-agent systems. These kinds of experiments are not cheap to conduct and the overreliance on external providers may become an increasingly glaring flaw in the future.

4.3 A critique of the experimental Setup

The design of the experiment itself was another important part of the entire project. In parallel with the conceptualization of the architecture, there was a reflection on how to put them to the test. The final arrangement proved to be quite successful at highlighting some of their characteristics. However, considering the observed results, it may be useful to reexamine the choices and comment briefly upon them. The objective here will not be to undermine the study itself, but to reflect on its methodologies in order to improve them in future works.

4.3.1 Prisoner's Dilemma: too simple?

First we need to consider the main focus of investigation: the Prisoner's Dilemma. The available literature seems unanimous in underlining the expressive capabilities of this simple game. Years of publications proved its versatility in many scenarios, making it a good model for simple social negotiation. However, in the case of our experiment, an argument could be made against its simplicity. While offering many views for interpretation, the final dilemma consists in a binary choice. It is arguable that such a limited amount of options may have acted as a bottleneck for all experimental features. While the results show clear patterns between the two types of agents, this idea can not be completely disproven. Ultimately the choice of the game itself may have been insufficient to capture the nuance expressed by both architectures. This does not invalidate the work done, which still provides ground for discussion. It simply means, in a direct analogy to behavioural studies, that the Prisoner's Dilemma must be interpreted as an introductory experiment. To further shed light on how LLM agent reason, even more complex experiments must be set up, and this study showed that game theory can be a good framework to design them.

4.3.2 Parameter choices

Having a quick glance at the results immediately highlights a difference in the importance of the observed experimental features. The study revolved around testing the agents on various combinations of conditions: Attitude, Opponent Strategy, Game Type, Number of Rounds, and underlying Model. From the inception of the experiment it was expected for each architecture to be influenced more by some parameters than others. More surprising was the fact that some of these almost did not matter at all. The best example is the number of rounds. A cause for this apparent irrelevance can be found in the chosen metrics, which do not reflect differences between longer and shorter games. Another explanation can be the choice of values: perhaps a difference of only 10 round is not sufficient to elicit a change in behaviour. This can be further confirmed by previous literature, where we often see games go up to 100 rounds or more. However, due to cost limitations concerning the Debate-based agent, experiments of such lengths could not be carried out. The effect of game type is another noticeable absent in most analyses. Aside from its relevance in the context of Tool-based agents with Nash tools, most LLM players seemed unaffected by it. This is peculiar but not too surprising, considering most LLM systems naturally struggle with attributing meaning to numerical values compared to words. This would also explain why all agents seem to pay more attention to text-based information such as opponent choices or attitude suggestions. While creating an experiment where every aspect mattered would have been ideal, some of these shortcomings in experimental design carry their own meaning and will be useful in future iterations.

4.4 Result Interpretation

Having provided an overview of all factors that may have affected the experiments, it is finally time to discuss the results properly. This discussion will revolve around the trends observed in the metrics, trying to give explanations to the behaviours expressed by the two examined architectures.

4.4.1 Addressing the Hypotheses

With all the experiments properly run the next step is reviewing the hypotheses proposed at the end of P. II, Ch. 2. As we will see, some proved to be accurate, while others seemed to miss the mark.

Hypothesis HT1. This hypothesis was confirmed by the experiments, which showed that tool suggestion takes priority over every other factor concerning the game. In fact, in Tool-based agents, cooperation rate was extremely high when *attitude*='cooperate' and almost zero when *attitude*='defect'.

Hypothesis HT2. Another hypothesis validated by the experiments, which proved that tools are the best available option to infuse LLM agents with symbolic knowledge. When guided by a Nash tool, the agent managed to properly recognize the game type and acting accordingly. There is, however, an issue with this. One may argue that the agent simply delegated the tool to find the best decision, rather than taking it by itself. While there may be truth to this, the important part is recognizing the agent's capability to correctly invoke the tool, which proved to be the true optimal choice.

Hypothesis HT3. Tool-based agents successfully confirms the third hypothesis by showing improvements in all questions concerning payoff related terms. Specifically, the highest improvement, compared to baseline, was achieved in spotting the minimum. This prompts further investigations into how additional tool context enhances agent reasoning. On a more negative note, every question unrelated to payoff aspects saw its performance degrading, which seems to suggest a confusing effect of the tool.

Hypothesis HD1. While there may be some hints of truth, there is no conclusive information to validate this claim. Both architectures seem to exhibit almost one-direction strategies. This can be also attributed to choice of opponent behaviour. Perhaps, Debate-based agents would have exhibited a more elaborate behaviour facing a more dynamic player. As things stand, both considering SFEM and feature importance analyses, we can not conclude that those agents display more complex behaviour.

Hypothesis HD2. This hypothesis was partially disproven when it comes to retaliation, but was mildly correct for forgiveness. While the expectation was for Debate-based agents to take more into account the opponent's behaviour, the results do not seem to fully support this claim. Overall these agents seem more prone to forgiving and the chance for it oddly increases when suggested defecting. On the side of retaliation nothing significant is noticed compared to the other agent types. Even here, the choice of more elaborate opponent strategies may have helped to clarify the situation.

Hypothesis HD3. This last hypothesis was completely disproven. Debate-based agents show the lowest score on opponent anticipation. This can be attributed to a general confusion of the entire system, but there may be a deeper meaning behind it. As shown by the results, the prediction seems lower when considering always cooperative opponents. This seems to imply that Debate-based agents seemed quite suspicious of their partners and were often anticipating a defection regardless of their apparent loyalty. As agents tend to validate each other, this inaccurate feedback may have increased minor suspicions into serious concerns.

4.4.2 Understanding Tool-Based Behaviour

The first architecture analysed was the Tool-based agent. When designing this typology of agents, the objective was clear: augmenting the SotA LLM architecture with external units to provide them with pieces of symbolic reasoning. In this regard, the objective can be considered achieved. As observed, the agent was capable of correctly invoking the tool and obtain useful suggestions, which it immediately put to use. Not only that, agents that used tools showed a more correct understanding of all aspects that concern their tool usage. This phenomenon is particularly fascinating since it shows that tools are not just an external unit that enhances the model's decisions. Under these premises, tools seem to help the agents focus more on aspects they would otherwise neglect simply by requiring said knowledge to use them. This can be attributed to the enhanced context each tool carries with itself. In a parallel with human cognition we can see a similar phenomenon. A tool (e.g. a hammer) is not just an object in the void: its features tell more to the user than just its potential use. They also give context on how it could interact with other objects (e.g. nails), extending the focus to them too. The potential for these unexpected consequences of tool usage has yet to be explored, but adds another layer of importance to their adoption.

The experiments, however, also highlighted some critical aspects of the tool-calling practice. Specifically, LLM agents appear to be particularly reliant on information provided by the tool. This is not surprising, considering that part of their fine-tuning was oriented at aligning them with the output of these modules. However, this phenomenon may come with its own set of problems. The first is the immediate necessity of making the functions the tool are based upon as reliable as possible. As agents trust the results almost blindly, this requirement is a baseline to guarantee a proper functioning. The next issue is specifically related to the usage of tools as suggestion systems. Observing the results we can conclude that such possibility seems hard to implement. LLM agents do not seem to elaborate much upon explicit suggestions, but instead take them at face value. We may speculate that a more careful and indirect phrasing of the output may help in this regard, but that may trigger another concern. If the result is considered inconclusive, the agent may be tempted to call the tool again, spiralling into an invocation loop. Additionally, it is unclear how an agent managing multiple tools would react after receiving many conflicting suggestions at the same time. These considerations hinder the idea of creating "soft" tools that influence the thought process of the agent without completely overriding it.

4.4.3 Understanding Debate-Based Behaviour

The second architecture presented in this study is the Debate-based agent. Being the most innovative between the two, the results were certainly less predicatable but potentially more interesting. Examining the performances, this agent underperformed most of its tasks. It provided significantly worse responses on comprehension questions and consistently subverted the suggested attitude. On top of this, it seemed more prone to defection and overall suspicious of its opponent. One might say that having more "voices inside its head" made it act erratically and there may be some truth to it. On paper, the idea of letting two separate agents debate the solution together was sound. The underlying assumption underestimated the confusion effect the two agents clearly had on each other. A free and unregulated conversation between two LLMs, joined with their appeasing behaviour, may let incomprehensions snowball into wrong decisions. This produces the opposite of the intended effect and can be attributed to a lack of governance in how the conversation flow was structured. We can speculate that a more structured back and forth, perhaps under the supervision of a judge agent, may help improve the results and bring out the potential of this architecture. In this sense, an appropriate study on this topic alone may be necessary.

The results are not, however, a complete failure. Some redeeming qualities can be found, especially with the goal of applying these architectures to human behaviour simulation. Humans are not rational decision makers, and the unpredictable nature of this type of agents, if properly governed, may be useful to capture these aspects. Debate-based agents seemed to be guided by the opponent's choice and tried to exploit cooperativeness, while punishing defection. This, combined with the rejection of the assistant's suggestion, seems to suggest a strategy based more on impulse rather than logic. It is not surprising, since LLMs are known for taking up the irrationality of humans together with their speech pattern. The extent to which the proposed architecture seems to enhance this form of behaviour is fascinating and may be used as a model to investigate other aspects of human irrationality.

Conclusions

Considerations on the project

With Part I and II coming to a close, we can finally conclude this dissertation by putting everything together into one perspective. This thesis started by presenting the potential of modern Artificial Intelligence systems in the field of negotiation, highlighting their growing relevance. Starting from this premise the next step was to determine a series of scenarios that could serve as testing grounds for these systems. In this context, game theory proved to be a valid proposal, as highlighted by the available literature. Then, we delved into the current state-of-the-art technologies which could apply to this sector, such as LLMs and their combination into larger multi-agent systems. With this information we approached the available literature on the topic to locate currently unexplored areas. Noticing a lack of investigation into architectural comparison, a series of experiments was designed on this topic.

The experimental setup joined all the previously described elements to carry out its assessments. The presented negotiation setting was modelled with a staple of game theory: the Prisoner's Dilemma. Two types of player agents were designed an implemented inside a dedicated framework to tackle this task. First a Tool-based agent, which implemented a small neuro-symbolic approach by joining LLM capabilities with an external logic-based module. Second a Debate-based agent, where to LLM units: decision-maker and assistant, discussed the situation trying to figure out the best solution. Experiments were carried out under a variety of conditions, most notable of which was the attitude which represented external interference in the overall decision process.

The results were in some cases expected and in other surprising. Tool-based agents showed extreme adherence with respect to the tool suggestion. This lead them to make proper decisions when the tool took into account the problem elements, and less accurate otherwise. Debate-based agents instead behaved more erratically, enhancing suspicion inside the agent. This generated a bizarre behaviour in which the decision maker more often than not did the opposite of what it was suggested to. Overall, the first showed a more well-rounded understanding of the scenarios when it came to direct comprehension.

Seeing these results we may conclude that equipping agents with tools seems to be the safest choice to enhance their reasoning capabilities. This, however, comes at the cost of limiting their innate behaviour as more and more decisions are attributed to logic-driven modules. Such design choice could indeed represent an obstacle in modelling human behaviour, which is proven to be frequently irrational. In this context, more debate driven architectures could become handy, granted that conversation flow is precisely regulated. Considering this, we may be moving towards even more complex architectures, perhaps featuring multiple "expert" agents, each equipped with its own set of tools. Joining these two separate approaches may be the key to achieving truly advanced reasoning patterns.

Future Work

As highlighted in the discussion chapter, this project, while effective, was plagued by limitations. This meant that many ideas that were originally thought of did not end up being implemented for the final experiments. In other cases it just signified that resource constraints impeded the execution of more ample tests. The first thing future developments of this project have to address is related to this last issue. While it is undeniable that designing complex architectures may make the entire system better, it can not be overstated how crucial the impact of the underlying LLMs is. Any of the presented structures depends heavily on the model that works beneath it. Thus, it is important to further investigate the effects of more sophisticated and diversified models, to assess the claims of this thesis and potentially make new ones. This would indeed come at a cost either towards LLM API providers or for more powerful resource to run locally. While the first option may guarantee the availability of cutting edge services, the current overreliance on few centralized providers is becoming concerning.

Lastly we move onto some of the concepts that were scrapped but could provide a good starting point for future work. Thanks to the relatively modular framework that was designed, implementing most of these ideas will not be too problematic. Testing additional architectures would be a great starting point. Specifically the effects of an agent equipped with multiple tools, each providing guidance on different aspects of the game. Another could see a council of multiple agents examining the situation, perhaps in a more organized setting than the presented Debate-based agent. A further addition could see the introduction of opponents with more elaborate strategies, which could actually challenge the reasoning of the LLM agents. With enough types of agents the end goal would be to create a modern version of Axelrod's tournament, featuring both logic and LLM players. After this it would be finally time to leave behind the Prisoner's Dilemma and explore more complex games such as beauty contests and public goods games. Both of these could provide further insight into the agent's reasoning, as they feature more fine-grained forms of decision-making. However, thanks to game theory, the list of scenarios is almost endless. The final goal would be to properly model real-life situations through these theoretical tools, finally bridging the gap between the initial intentions of this work and the actual research.

Whatever future instalments entail, the potential for this kind of studies is vast and its implications both fascinating and challenging. Only one thing is clear: this work represents a small part in a journey that it is only at its beginning.

Acknowledgements

The project behind this thesis would not have been possible without the support of the two institutions that hosted me throughout its development: the University of Bologna and the University of Birmingham. Specifically, I would like to thank my supervisor, Prof. Roberta Calegari, for providing guidance in organizing the project and writing this document. I extend my thanks to Prof. Paolo Torroni, who added valuable insight into the experimental setup. I additionally thank Prof. Christian Arnold, who made the experience behind this possible in the first place and helped me develop these experiments. My thanks go to the entire CAIG team, who worked with me during my experience abroad.

I should also thank my parents and relatives for encouraging me during these years of study and for loving me over my entire life. Lastly, I thank my friends, whose laughters and affection accompanied me during this long period. Specifically, I need to thank Gabriele, who shared this long and ambitious journey with me.

To all of you: Grazie.

Bibliography

[1] *Ag2 Homepage*. https://docs.ag2.ai/latest/.

[2] J. Ainslie et al. *GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints*. 2023. arXiv: 2305.13245 [cs.CL]. URL: https://arxiv.org/abs/2305.13245.

[3] *Autogen Homepage*. https://microsoft.github.io/autogen/stable/index.html.

[4] R. Axelrod. *The Evolution of Cooperation*. Basic Books, 2025. ISBN: 9781541606944. URL: https://books.google.it/books?id=dadkEQAAQBAJ.

[5] E. M. Bender et al. 'On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? □'. In: *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*. FAccT '21. Virtual Event, Canada: Association for Computing Machinery, 2021, pp. 610–623. ISBN: 9781450383097. DOI: 10.1145/3442188.3445922. URL: https://doi.org/10.1145/3442188.3445922.

[6] C. Camerer. 'Behavioral Game Theory'. In: *Encyclopedia of Evolutionary Psychological Science* (2021). URL: https://api.semanticscholar.org/CorpusID:148985378.

[7] *CrewAI Homepage*. https://docs.crewai.com/.

[8] J. Duan et al. *GTBench: Uncovering the Strategic Reasoning Limitations of LLMs via Game-Theoretic Evaluations*. 2024. arXiv: 2402.12348 [cs.CL]. URL: https://arxiv.org/abs/2402.12348.

[9] N. Fontana, F. Pierri and L. M. Aiello. *Nicer Than Humans: How do Large Language Models Behave in the Prisoner's Dilemma?* 2024. arXiv: 2406.13605 [cs.CY]. URL: https://arxiv.org/abs/2406.13605.

[10] W. Gan et al. *Mixture of Experts (MoE): A Big Data Perspective*. 2025. arXiv: 2501.16352 [cs.LG]. URL: https://arxiv.org/abs/2501.16352.

[11] N. Herr et al. *Are Large Language Models Strategic Decision Makers? A Study of Performance and Bias in Two-Player Non-Zero-Sum Games*. 2024. arXiv: 2407.04467 [cs.AI]. URL: https://arxiv.org/abs/2407.04467.

[12] *Introducing GPT-OSS*. https://openai.com/it-IT/index/introducing-gpt-oss/. Accessed: Aug , 2025.

[13] V. Knight and J. Campbell. 'Nashpy: A Python library for the computation of Nash equilibria'. In: *Journal of Open Source Software* 3 (Oct. 2018), p. 904. DOI: 10.21105/joss.00904.

[14] *LangGraph Homepage*. https://www.langchain.com/langgraph.

[15] S. E. Lu. 'Game-theory behaviour of large language models: The case of Keynesian beauty contests'. In: *Economics and Business Review* 11.2 (2025), pp. 119–148. DOI: 10.18559/ebr.2025.2.2182. URL: https://doi.org/10.18559/ebr.2025.2.2182.

[16] *Mistral 3.1 Homepage*. https://mistral.ai/news/mistral-small-3-1. Accessed: Mar 17, 2025.

[17] A. Orland and K. Takemoto. *Playing Prisoner's Dilemma Games with a Large Language Model*. Jan. 2025. DOI: 10.2139/ssrn.5716903.

[18] I. Reynolds and B. Jensen. *Machine Learning Meets War Termination: Using AI to Explore Peace Scenarios in Ukraine*. Tech. rep. Center for Strategic and International Studies (CSIS), 2025. URL: http://www.jstor.org/stable/resrep68080 (visited on 02/02/2026).

[19] J. Romero and Y. Rosokha. 'Constructing strategies in the indefinitely repeated prisoner's dilemma game'. In: *European Economic Review* 104 (2018), pp. 185–219. ISSN: 0014-2921. DOI: https://doi.org/10.1016/j.euroecorev.2018.02.008. URL: https://www.sciencedirect.com/science/article/pii/S0014292118300394.

[20] T. Schick et al. *Toolformer: Language Models Can Teach Themselves to Use Tools*. 2023. arXiv: 2302.04761 [cs.CL]. URL: https://arxiv.org/abs/2302.04761.

[21] S. Schulhoff et al. *The Prompt Report: A Systematic Survey of Prompt Engineering Techniques*. 2025. arXiv: 2406.06608 [cs.CL]. URL: https://arxiv.org/abs/2406.06608.

[22] N. Shazeer et al. *Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer*. 2017. arXiv: 1701.06538 [cs.LG]. URL: https://arxiv.org/abs/1701.06538.

[23] A. Smit et al. *Should we be going MAD? A Look at Multi-Agent Debate Strategies for LLMs*. 2024. arXiv: 2311.17371 [cs.CL]. URL: https://arxiv.org/abs/2311.17371.

[24] M. H. Tessler et al. 'AI can help humans find common ground in democratic deliberation'. In: *Science* 386.6719 (2024), eadq2852. DOI: 10.1126/science.adq2852. eprint: https://www.science.org/doi/pdf/10.1126/science.adq2852. URL: https://www.science.org/doi/abs/10.1126/science.adq2852.

[25] *The Big LLM Architecture Comparison*. https://magazine.sebastianraschka.com/p/the-big-llm-architecture-comparison. Accessed: Jul 19, 2025.

[26] *The Llama 4 herd: The beginning of a new era of natively multimodal AI innovation*. https://ai.meta.com/blog/llama-4-multimodal-intelligence/. Accessed: Apr 5, 2025.

[27] A. Vaswani et al. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762.

[28] *What are AI agents?* https://www.ibm.com/think/topics/ai-agents.

[29] *What are Tools?* https://huggingface.co/learn/agents-course/en/unit1/tools.

[30] K. Zheng, J. Zhou and H. Wang. *Beyond Nash Equilibrium: Bounded Rationality of LLMs and humans in Strategic Decision-making*. 2025. arXiv: 2506.09390 [cs.AI]. URL: https://arxiv.org/abs/2506.09390.

[31] S. Zhu et al. *The Automated but Risky Game: Modeling and Benchmarking Agent-to-Agent Negotiations and Transactions in Consumer Markets*. 2025. arXiv: 2506.00073 [cs.AI]. URL: https://arxiv.org/abs/2506.00073.

[32] V. Zwass. *Encyclopedia Britannica*. https://www.britannica.com/technology/software-agent. May 29, 2023.

Appendix A

Experiment Results

Table A.1: Features of all games

	Model	Arch	OppStr	Att	NR	Game
0	gpt-4o-mini	Plain_LLM	AlwaysCooperate	NA	10	Dilemma
1	gpt-4o-mini	Plain_LLM	AlwaysCooperate	NA	10	Delight
2	gpt-4o-mini	Plain_LLM	AlwaysCooperate	NA	10	Confusion
3	gpt-4o-mini	Plain_LLM	AlwaysCooperate	NA	20	Dilemma
4	gpt-4o-mini	Plain_LLM	AlwaysCooperate	NA	20	Delight
5	gpt-4o-mini	Plain_LLM	AlwaysCooperate	NA	20	Confusion
6	gpt-4o-mini	Plain_LLM	AlwaysDefect	NA	10	Dilemma
7	gpt-4o-mini	Plain_LLM	AlwaysDefect	NA	10	Delight
8	gpt-4o-mini	Plain_LLM	AlwaysDefect	NA	10	Confusion
9	gpt-4o-mini	Plain_LLM	AlwaysDefect	NA	20	Dilemma
10	gpt-4o-mini	Plain_LLM	AlwaysDefect	NA	20	Delight
11	gpt-4o-mini	Plain_LLM	AlwaysDefect	NA	20	Confusion
12	gpt-4o-mini	Plain_LLM	TitForTat	NA	10	Dilemma
13	gpt-4o-mini	Plain_LLM	TitForTat	NA	10	Delight
14	gpt-4o-mini	Plain_LLM	TitForTat	NA	10	Confusion

Continued on next page

Table A.1: Features of all games

	Model	Arch	OppStr	Att	NR	Game
15	gpt-4o-mini	Plain_LLM	TitForTat	NA	20	Dilemma
16	gpt-4o-mini	Plain_LLM	TitForTat	NA	20	Delight
17	gpt-4o-mini	Plain_LLM	TitForTat	NA	20	Confusion
18	gpt-4o-mini	Tool	AlwaysCooperate	cooperate	10	Dilemma
19	gpt-4o-mini	Tool	AlwaysCooperate	cooperate	10	Delight
20	gpt-4o-mini	Tool	AlwaysCooperate	cooperate	10	Confusion
21	gpt-4o-mini	Tool	AlwaysCooperate	cooperate	20	Dilemma
22	gpt-4o-mini	Tool	AlwaysCooperate	cooperate	20	Delight
23	gpt-4o-mini	Tool	AlwaysCooperate	cooperate	20	Confusion
24	gpt-4o-mini	Tool	AlwaysCooperate	defect	10	Dilemma
25	gpt-4o-mini	Tool	AlwaysCooperate	defect	10	Delight
26	gpt-4o-mini	Tool	AlwaysCooperate	defect	10	Confusion
27	gpt-4o-mini	Tool	AlwaysCooperate	defect	20	Dilemma
28	gpt-4o-mini	Tool	AlwaysCooperate	defect	20	Delight
29	gpt-4o-mini	Tool	AlwaysCooperate	defect	20	Confusion
30	gpt-4o-mini	Tool	AlwaysCooperate	compute	10	Dilemma
31	gpt-4o-mini	Tool	AlwaysCooperate	compute	10	Delight
32	gpt-4o-mini	Tool	AlwaysCooperate	compute	10	Confusion
33	gpt-4o-mini	Tool	AlwaysCooperate	compute	20	Dilemma
34	gpt-4o-mini	Tool	AlwaysCooperate	compute	20	Delight
35	gpt-4o-mini	Tool	AlwaysCooperate	compute	20	Confusion
36	gpt-4o-mini	Tool	AlwaysDefect	cooperate	10	Dilemma
37	gpt-4o-mini	Tool	AlwaysDefect	cooperate	10	Delight
38	gpt-4o-mini	Tool	AlwaysDefect	cooperate	10	Confusion
39	gpt-4o-mini	Tool	AlwaysDefect	cooperate	20	Dilemma

Continued on next page

Table A.1: Features of all games

	Model	Arch	OppStr	Att	NR	Game
40	gpt-4o-mini	Tool	AlwaysDefect	cooperate	20	Delight
41	gpt-4o-mini	Tool	AlwaysDefect	cooperate	20	Confusion
42	gpt-4o-mini	Tool	AlwaysDefect	defect	10	Dilemma
43	gpt-4o-mini	Tool	AlwaysDefect	defect	10	Delight
44	gpt-4o-mini	Tool	AlwaysDefect	defect	10	Confusion
45	gpt-4o-mini	Tool	AlwaysDefect	defect	20	Dilemma
46	gpt-4o-mini	Tool	AlwaysDefect	defect	20	Delight
47	gpt-4o-mini	Tool	AlwaysDefect	defect	20	Confusion
48	gpt-4o-mini	Tool	AlwaysDefect	compute	10	Dilemma
49	gpt-4o-mini	Tool	AlwaysDefect	compute	10	Delight
50	gpt-4o-mini	Tool	AlwaysDefect	compute	10	Confusion
51	gpt-4o-mini	Tool	AlwaysDefect	compute	20	Dilemma
52	gpt-4o-mini	Tool	AlwaysDefect	compute	20	Delight
53	gpt-4o-mini	Tool	AlwaysDefect	compute	20	Confusion
54	gpt-4o-mini	Tool	TitForTat	cooperate	10	Dilemma
55	gpt-4o-mini	Tool	TitForTat	cooperate	10	Delight
56	gpt-4o-mini	Tool	TitForTat	cooperate	10	Confusion
57	gpt-4o-mini	Tool	TitForTat	cooperate	20	Dilemma
58	gpt-4o-mini	Tool	TitForTat	cooperate	20	Delight
59	gpt-4o-mini	Tool	TitForTat	cooperate	20	Confusion
60	gpt-4o-mini	Tool	TitForTat	defect	10	Dilemma
61	gpt-4o-mini	Tool	TitForTat	defect	10	Delight
62	gpt-4o-mini	Tool	TitForTat	defect	10	Confusion
63	gpt-4o-mini	Tool	TitForTat	defect	20	Dilemma
64	gpt-4o-mini	Tool	TitForTat	defect	20	Delight

Continued on next page

Table A.1: Features of all games

	Model	Arch	OppStr	Att	NR	Game
65	gpt-4o-mini	Tool	TitForTat	defect	20	Confusion
66	gpt-4o-mini	Tool	TitForTat	compute	10	Dilemma
67	gpt-4o-mini	Tool	TitForTat	compute	10	Delight
68	gpt-4o-mini	Tool	TitForTat	compute	10	Confusion
69	gpt-4o-mini	Tool	TitForTat	compute	20	Dilemma
70	gpt-4o-mini	Tool	TitForTat	compute	20	Delight
71	gpt-4o-mini	Tool	TitForTat	compute	20	Confusion
72	gpt-4o-mini	Assistant	AlwaysCooperate	cooperate	10	Dilemma
73	gpt-4o-mini	Assistant	AlwaysCooperate	cooperate	10	Delight
74	gpt-4o-mini	Assistant	AlwaysCooperate	cooperate	10	Confusion
75	gpt-4o-mini	Assistant	AlwaysCooperate	cooperate	20	Dilemma
76	gpt-4o-mini	Assistant	AlwaysCooperate	cooperate	20	Delight
77	gpt-4o-mini	Assistant	AlwaysCooperate	cooperate	20	Confusion
78	gpt-4o-mini	Assistant	AlwaysCooperate	defect	10	Dilemma
79	gpt-4o-mini	Assistant	AlwaysCooperate	defect	10	Delight
80	gpt-4o-mini	Assistant	AlwaysCooperate	defect	10	Confusion
81	gpt-4o-mini	Assistant	AlwaysCooperate	defect	20	Dilemma
82	gpt-4o-mini	Assistant	AlwaysCooperate	defect	20	Delight
83	gpt-4o-mini	Assistant	AlwaysCooperate	defect	20	Confusion
84	gpt-4o-mini	Assistant	AlwaysCooperate	compute	10	Dilemma
85	gpt-4o-mini	Assistant	AlwaysCooperate	compute	10	Delight
86	gpt-4o-mini	Assistant	AlwaysCooperate	compute	10	Confusion
87	gpt-4o-mini	Assistant	AlwaysCooperate	compute	20	Dilemma
88	gpt-4o-mini	Assistant	AlwaysCooperate	compute	20	Delight
89	gpt-4o-mini	Assistant	AlwaysCooperate	compute	20	Confusion

Continued on next page

Table A.1: Features of all games

	Model	Arch	OppStr	Att	NR	Game
90	gpt-4o-mini	Assistant	AlwaysDefect	cooperate	10	Dilemma
91	gpt-4o-mini	Assistant	AlwaysDefect	cooperate	10	Delight
92	gpt-4o-mini	Assistant	AlwaysDefect	cooperate	10	Confusion
93	gpt-4o-mini	Assistant	AlwaysDefect	cooperate	20	Dilemma
94	gpt-4o-mini	Assistant	AlwaysDefect	cooperate	20	Delight
95	gpt-4o-mini	Assistant	AlwaysDefect	cooperate	20	Confusion
96	gpt-4o-mini	Assistant	AlwaysDefect	defect	10	Dilemma
97	gpt-4o-mini	Assistant	AlwaysDefect	defect	10	Delight
98	gpt-4o-mini	Assistant	AlwaysDefect	defect	10	Confusion
99	gpt-4o-mini	Assistant	AlwaysDefect	defect	20	Dilemma
100	gpt-4o-mini	Assistant	AlwaysDefect	defect	20	Delight
101	gpt-4o-mini	Assistant	AlwaysDefect	defect	20	Confusion
102	gpt-4o-mini	Assistant	AlwaysDefect	compute	10	Dilemma
103	gpt-4o-mini	Assistant	AlwaysDefect	compute	10	Delight
104	gpt-4o-mini	Assistant	AlwaysDefect	compute	10	Confusion
105	gpt-4o-mini	Assistant	AlwaysDefect	compute	20	Dilemma
106	gpt-4o-mini	Assistant	AlwaysDefect	compute	20	Delight
107	gpt-4o-mini	Assistant	AlwaysDefect	compute	20	Confusion
108	gpt-4o-mini	Assistant	TitForTat	cooperate	10	Dilemma
109	gpt-4o-mini	Assistant	TitForTat	cooperate	10	Delight
110	gpt-4o-mini	Assistant	TitForTat	cooperate	10	Confusion
111	gpt-4o-mini	Assistant	TitForTat	cooperate	20	Dilemma
112	gpt-4o-mini	Assistant	TitForTat	cooperate	20	Delight
113	gpt-4o-mini	Assistant	TitForTat	cooperate	20	Confusion
114	gpt-4o-mini	Assistant	TitForTat	defect	10	Dilemma

Continued on next page

Table A.1: Features of all games

	Model	Arch	OppStr	Att	NR	Game
115	gpt-4o-mini	Assistant	TitForTat	defect	10	Delight
116	gpt-4o-mini	Assistant	TitForTat	defect	10	Confusion
117	gpt-4o-mini	Assistant	TitForTat	defect	20	Dilemma
118	gpt-4o-mini	Assistant	TitForTat	defect	20	Delight
119	gpt-4o-mini	Assistant	TitForTat	defect	20	Confusion
120	gpt-4o-mini	Assistant	TitForTat	compute	10	Dilemma
121	gpt-4o-mini	Assistant	TitForTat	compute	10	Delight
122	gpt-4o-mini	Assistant	TitForTat	compute	10	Confusion
123	gpt-4o-mini	Assistant	TitForTat	compute	20	Dilemma
124	gpt-4o-mini	Assistant	TitForTat	compute	20	Delight
125	gpt-4o-mini	Assistant	TitForTat	compute	20	Confusion
126	ministral-8b	Plain_LLM	AlwaysCooperate	NA	10	Dilemma
127	ministral-8b	Plain_LLM	AlwaysCooperate	NA	10	Delight
128	ministral-8b	Plain_LLM	AlwaysCooperate	NA	10	Confusion
129	ministral-8b	Plain_LLM	AlwaysCooperate	NA	20	Dilemma
130	ministral-8b	Plain_LLM	AlwaysCooperate	NA	20	Delight
131	ministral-8b	Plain_LLM	AlwaysCooperate	NA	20	Confusion
132	ministral-8b	Plain_LLM	AlwaysDefect	NA	10	Dilemma
133	ministral-8b	Plain_LLM	AlwaysDefect	NA	10	Delight
134	ministral-8b	Plain_LLM	AlwaysDefect	NA	10	Confusion
135	ministral-8b	Plain_LLM	AlwaysDefect	NA	20	Dilemma
136	ministral-8b	Plain_LLM	AlwaysDefect	NA	20	Delight
137	ministral-8b	Plain_LLM	AlwaysDefect	NA	20	Confusion
138	ministral-8b	Plain_LLM	TitForTat	NA	10	Dilemma
139	ministral-8b	Plain_LLM	TitForTat	NA	10	Delight

Continued on next page

Table A.1: Features of all games

	Model	Arch	OppStr	Att	NR	Game
140	ministral-8b	Plain_LLM	TitForTat	NA	10	Confusion
141	ministral-8b	Plain_LLM	TitForTat	NA	20	Dilemma
142	ministral-8b	Plain_LLM	TitForTat	NA	20	Delight
143	ministral-8b	Plain_LLM	TitForTat	NA	20	Confusion
144	ministral-8b	Tool	AlwaysCooperate	cooperate	10	Dilemma
145	ministral-8b	Tool	AlwaysCooperate	cooperate	10	Delight
146	ministral-8b	Tool	AlwaysCooperate	cooperate	10	Confusion
147	ministral-8b	Tool	AlwaysCooperate	cooperate	20	Dilemma
148	ministral-8b	Tool	AlwaysCooperate	cooperate	20	Delight
149	ministral-8b	Tool	AlwaysCooperate	cooperate	20	Confusion
150	ministral-8b	Tool	AlwaysCooperate	defect	10	Dilemma
151	ministral-8b	Tool	AlwaysCooperate	defect	10	Delight
152	ministral-8b	Tool	AlwaysCooperate	defect	10	Confusion
153	ministral-8b	Tool	AlwaysCooperate	defect	20	Dilemma
154	ministral-8b	Tool	AlwaysCooperate	defect	20	Delight
155	ministral-8b	Tool	AlwaysCooperate	defect	20	Confusion
156	ministral-8b	Tool	AlwaysCooperate	compute	10	Dilemma
157	ministral-8b	Tool	AlwaysCooperate	compute	10	Delight
158	ministral-8b	Tool	AlwaysCooperate	compute	10	Confusion
159	ministral-8b	Tool	AlwaysCooperate	compute	20	Dilemma
160	ministral-8b	Tool	AlwaysCooperate	compute	20	Delight
161	ministral-8b	Tool	AlwaysCooperate	compute	20	Confusion
162	ministral-8b	Tool	AlwaysDefect	cooperate	10	Dilemma
163	ministral-8b	Tool	AlwaysDefect	cooperate	10	Delight
164	ministral-8b	Tool	AlwaysDefect	cooperate	10	Confusion

Continued on next page

Table A.1: Features of all games

	Model	Arch	OppStr	Att	NR	Game
165	ministral-8b	Tool	AlwaysDefect	cooperate	20	Dilemma
166	ministral-8b	Tool	AlwaysDefect	cooperate	20	Delight
167	ministral-8b	Tool	AlwaysDefect	cooperate	20	Confusion
168	ministral-8b	Tool	AlwaysDefect	defect	10	Dilemma
169	ministral-8b	Tool	AlwaysDefect	defect	10	Delight
170	ministral-8b	Tool	AlwaysDefect	defect	10	Confusion
171	ministral-8b	Tool	AlwaysDefect	defect	20	Dilemma
172	ministral-8b	Tool	AlwaysDefect	defect	20	Delight
173	ministral-8b	Tool	AlwaysDefect	defect	20	Confusion
174	ministral-8b	Tool	AlwaysDefect	compute	10	Dilemma
175	ministral-8b	Tool	AlwaysDefect	compute	10	Delight
176	ministral-8b	Tool	AlwaysDefect	compute	10	Confusion
177	ministral-8b	Tool	AlwaysDefect	compute	20	Dilemma
178	ministral-8b	Tool	AlwaysDefect	compute	20	Delight
179	ministral-8b	Tool	AlwaysDefect	compute	20	Confusion
180	ministral-8b	Tool	TitForTat	cooperate	10	Dilemma
181	ministral-8b	Tool	TitForTat	cooperate	10	Delight
182	ministral-8b	Tool	TitForTat	cooperate	10	Confusion
183	ministral-8b	Tool	TitForTat	cooperate	20	Dilemma
184	ministral-8b	Tool	TitForTat	cooperate	20	Delight
185	ministral-8b	Tool	TitForTat	cooperate	20	Confusion
186	ministral-8b	Tool	TitForTat	defect	10	Dilemma
187	ministral-8b	Tool	TitForTat	defect	10	Delight
188	ministral-8b	Tool	TitForTat	defect	10	Confusion
189	ministral-8b	Tool	TitForTat	defect	20	Dilemma

Continued on next page

Table A.1: Features of all games

	Model	Arch	OppStr	Att	NR	Game
190	ministral-8b	Tool	TitForTat	defect	20	Delight
191	ministral-8b	Tool	TitForTat	defect	20	Confusion
192	ministral-8b	Tool	TitForTat	compute	10	Dilemma
193	ministral-8b	Tool	TitForTat	compute	10	Delight
194	ministral-8b	Tool	TitForTat	compute	10	Confusion
195	ministral-8b	Tool	TitForTat	compute	20	Dilemma
196	ministral-8b	Tool	TitForTat	compute	20	Delight
197	ministral-8b	Tool	TitForTat	compute	20	Confusion
198	ministral-8b	Assistant	AlwaysCooperate	cooperate	10	Dilemma
199	ministral-8b	Assistant	AlwaysCooperate	cooperate	10	Delight
200	ministral-8b	Assistant	AlwaysCooperate	cooperate	10	Confusion
201	ministral-8b	Assistant	AlwaysCooperate	cooperate	20	Dilemma
202	ministral-8b	Assistant	AlwaysCooperate	cooperate	20	Delight
203	ministral-8b	Assistant	AlwaysCooperate	cooperate	20	Confusion
204	ministral-8b	Assistant	AlwaysCooperate	defect	10	Dilemma
205	ministral-8b	Assistant	AlwaysCooperate	defect	10	Delight
206	ministral-8b	Assistant	AlwaysCooperate	defect	10	Confusion
207	ministral-8b	Assistant	AlwaysCooperate	defect	20	Dilemma
208	ministral-8b	Assistant	AlwaysCooperate	defect	20	Delight
209	ministral-8b	Assistant	AlwaysCooperate	defect	20	Confusion
210	ministral-8b	Assistant	AlwaysCooperate	compute	10	Dilemma
211	ministral-8b	Assistant	AlwaysCooperate	compute	10	Delight
212	ministral-8b	Assistant	AlwaysCooperate	compute	10	Confusion
213	ministral-8b	Assistant	AlwaysCooperate	compute	20	Dilemma
214	ministral-8b	Assistant	AlwaysCooperate	compute	20	Delight

Continued on next page

Table A.1: Features of all games

	Model	Arch	OppStr	Att	NR	Game
215	ministral-8b	Assistant	AlwaysCooperate	compute	20	Confusion
216	ministral-8b	Assistant	AlwaysDefect	cooperate	10	Dilemma
217	ministral-8b	Assistant	AlwaysDefect	cooperate	10	Delight
218	ministral-8b	Assistant	AlwaysDefect	cooperate	10	Confusion
219	ministral-8b	Assistant	AlwaysDefect	cooperate	20	Dilemma
220	ministral-8b	Assistant	AlwaysDefect	cooperate	20	Delight
221	ministral-8b	Assistant	AlwaysDefect	cooperate	20	Confusion
222	ministral-8b	Assistant	AlwaysDefect	defect	10	Dilemma
223	ministral-8b	Assistant	AlwaysDefect	defect	10	Delight
224	ministral-8b	Assistant	AlwaysDefect	defect	10	Confusion
225	ministral-8b	Assistant	AlwaysDefect	defect	20	Dilemma
226	ministral-8b	Assistant	AlwaysDefect	defect	20	Delight
227	ministral-8b	Assistant	AlwaysDefect	defect	20	Confusion
228	ministral-8b	Assistant	AlwaysDefect	compute	10	Dilemma
229	ministral-8b	Assistant	AlwaysDefect	compute	10	Delight
230	ministral-8b	Assistant	AlwaysDefect	compute	10	Confusion
231	ministral-8b	Assistant	AlwaysDefect	compute	20	Dilemma
232	ministral-8b	Assistant	AlwaysDefect	compute	20	Delight
233	ministral-8b	Assistant	AlwaysDefect	compute	20	Confusion
234	ministral-8b	Assistant	TitForTat	cooperate	10	Dilemma
235	ministral-8b	Assistant	TitForTat	cooperate	10	Delight
236	ministral-8b	Assistant	TitForTat	cooperate	10	Confusion
237	ministral-8b	Assistant	TitForTat	cooperate	20	Dilemma
238	ministral-8b	Assistant	TitForTat	cooperate	20	Delight
239	ministral-8b	Assistant	TitForTat	cooperate	20	Confusion

Continued on next page

Table A.1: Features of all games

	Model	Arch	OppStr	Att	NR	Game
240	ministral-8b	Assistant	TitForTat	defect	10	Dilemma
241	ministral-8b	Assistant	TitForTat	defect	10	Delight
242	ministral-8b	Assistant	TitForTat	defect	10	Confusion
243	ministral-8b	Assistant	TitForTat	defect	20	Dilemma
244	ministral-8b	Assistant	TitForTat	defect	20	Delight
245	ministral-8b	Assistant	TitForTat	defect	20	Confusion
246	ministral-8b	Assistant	TitForTat	compute	10	Dilemma
247	ministral-8b	Assistant	TitForTat	compute	10	Delight
248	ministral-8b	Assistant	TitForTat	compute	10	Confusion
249	ministral-8b	Assistant	TitForTat	compute	20	Dilemma
250	ministral-8b	Assistant	TitForTat	compute	20	Delight
251	ministral-8b	Assistant	TitForTat	compute	20	Confusion

Table A.2: Comprehension Metricss of all games

	CR1	CR2	CR3M	CR3m	CR4	MR1	MR2	MR3M	MR3m	MR4
0	0.90	1.00	1.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
1	0.90	1.00	1.00	0.10	1.00	0.00	0.00	0.00	0.00	0.00
2	0.90	1.00	1.00	1.00	1.00	0.00	0.00	0.00	0.00	0.00
3	0.95	1.00	0.90	0.10	1.00	0.00	0.00	0.10	0.10	0.00
4	0.95	1.00	0.90	0.00	1.00	0.00	0.00	0.10	0.10	0.00
5	0.95	0.95	0.85	0.85	1.00	0.00	0.00	0.15	0.15	0.00
6	0.90	1.00	0.90	0.90	0.20	0.00	0.00	0.10	0.10	0.00
7	0.90	1.00	0.90	0.90	0.00	0.00	0.00	0.10	0.10	0.00
8	0.90	1.00	1.00	1.00	0.30	0.00	0.00	0.00	0.00	0.00

Continued on next page

Table A.2: Comprehension Metricss of all games

	CR1	CR2	CR3M	CR3m	CR4	MR1	MR2	MR3M	MR3m	MR4
9	0.95	1.00	0.90	0.85	0.10	0.00	0.00	0.10	0.10	0.00
10	0.95	0.95	0.90	0.90	0.00	0.00	0.05	0.10	0.10	0.00
11	0.95	1.00	0.90	0.90	0.10	0.00	0.00	0.10	0.10	0.00
12	0.90	1.00	0.80	0.00	1.00	0.00	0.00	0.20	0.20	0.00
13	0.90	1.00	0.80	0.10	1.00	0.00	0.00	0.20	0.20	0.00
14	0.90	1.00	1.00	1.00	1.00	0.00	0.00	0.00	0.00	0.00
15	0.95	1.00	0.85	0.00	1.00	0.00	0.00	0.15	0.15	0.00
16	0.95	0.95	0.85	0.05	1.00	0.00	0.05	0.15	0.15	0.00
17	0.95	1.00	0.95	0.95	1.00	0.00	0.00	0.05	0.05	0.00
18	0.90	1.00	1.00	0.20	1.00	0.00	0.00	0.00	0.00	0.00
19	0.90	1.00	1.00	0.60	1.00	0.00	0.00	0.00	0.00	0.00
20	0.90	1.00	0.90	0.90	1.00	0.00	0.00	0.10	0.10	0.00
21	0.95	1.00	1.00	0.35	1.00	0.00	0.00	0.00	0.00	0.00
22	0.95	1.00	1.00	0.60	1.00	0.00	0.00	0.00	0.00	0.00
23	0.95	1.00	1.00	1.00	1.00	0.00	0.00	0.00	0.00	0.00
24	0.10	1.00	0.90	0.90	0.90	0.00	0.00	0.00	0.00	0.00
25	0.00	1.00	1.00	1.00	0.20	0.00	0.00	0.00	0.00	0.00
26	0.00	1.00	1.00	0.70	0.70	0.00	0.00	0.00	0.00	0.00
27	0.05	1.00	0.95	0.90	0.45	0.00	0.00	0.05	0.05	0.00
28	0.00	1.00	0.90	0.95	1.00	0.00	0.00	0.05	0.05	0.00
29	0.00	1.00	0.90	0.60	0.95	0.00	0.00	0.10	0.10	0.05
30	0.00	1.00	0.20	0.90	1.00	0.00	0.00	0.00	0.00	0.00
31	0.90	1.00	0.90	0.90	1.00	0.00	0.00	0.10	0.10	0.00
32	0.40	1.00	0.50	0.50	0.30	0.00	0.00	0.00	0.00	0.00
33	0.00	1.00	0.05	0.90	0.75	0.00	0.00	0.05	0.05	0.00

Continued on next page

Table A.2: Comprehension Metricss of all games

	CR1	CR2	CR3M	CR3m	CR4	MR1	MR2	MR3M	MR3m	MR4
34	0.95	1.00	0.95	0.90	1.00	0.00	0.00	0.05	0.05	0.00
35	0.60	1.00	0.40	0.40	0.20	0.00	0.00	0.10	0.10	0.00
36	0.00	1.00	1.00	1.00	0.40	0.00	0.00	0.00	0.00	0.00
37	0.50	1.00	0.90	0.90	0.20	0.00	0.00	0.10	0.10	0.00
38	0.40	1.00	0.90	0.90	0.70	0.00	0.00	0.10	0.10	0.00
39	0.05	1.00	1.00	0.95	0.55	0.00	0.00	0.00	0.00	0.00
40	0.25	1.00	0.95	0.95	0.45	0.00	0.00	0.05	0.05	0.00
41	0.25	1.00	1.00	0.65	0.75	0.00	0.00	0.00	0.00	0.00
42	0.90	1.00	0.80	0.90	0.50	0.00	0.00	0.00	0.00	0.00
43	0.90	1.00	0.90	0.80	1.00	0.00	0.00	0.10	0.10	0.00
44	0.90	1.00	0.40	0.60	0.90	0.00	0.00	0.20	0.20	0.00
45	0.95	1.00	0.55	0.70	0.85	0.00	0.00	0.00	0.00	0.00
46	0.95	1.00	0.95	0.95	1.00	0.00	0.00	0.05	0.05	0.00
47	0.95	1.00	0.60	0.80	1.00	0.00	0.00	0.10	0.10	0.00
48	0.90	1.00	0.10	0.90	1.00	0.00	0.00	0.10	0.10	0.00
49	0.80	1.00	1.00	1.00	0.10	0.00	0.00	0.00	0.00	0.00
50	0.40	1.00	0.70	0.70	0.30	0.00	0.00	0.00	0.00	0.00
51	0.95	1.00	0.10	0.85	1.00	0.00	0.00	0.00	0.00	0.00
52	0.35	1.00	1.00	1.00	0.30	0.00	0.00	0.00	0.00	0.00
53	0.45	1.00	0.55	0.55	0.25	0.00	0.00	0.00	0.00	0.00
54	0.90	1.00	1.00	0.40	1.00	0.00	0.00	0.00	0.00	0.00
55	0.90	1.00	1.00	1.00	1.00	0.00	0.00	0.00	0.00	0.00
56	0.90	1.00	0.90	0.90	1.00	0.00	0.00	0.10	0.10	0.00
57	0.95	1.00	0.95	0.40	1.00	0.00	0.00	0.00	0.00	0.00
58	0.95	1.00	0.95	0.55	1.00	0.00	0.00	0.05	0.05	0.00

Continued on next page

Table A.2: Comprehension Metricss of all games

	CR1	CR2	CR3M	CR3m	CR4	MR1	MR2	MR3M	MR3m	MR4
59	0.95	1.00	0.95	0.95	1.00	0.00	0.00	0.05	0.05	0.00
60	0.90	1.00	0.90	0.90	0.20	0.00	0.00	0.00	0.00	0.00
61	0.90	0.90	1.00	1.00	0.60	0.00	0.10	0.00	0.00	0.00
62	0.90	1.00	0.80	0.90	0.40	0.00	0.00	0.00	0.00	0.00
63	0.95	1.00	0.90	0.95	0.10	0.00	0.00	0.05	0.05	0.00
64	0.95	1.00	0.85	0.80	0.35	0.00	0.00	0.15	0.15	0.00
65	0.90	1.00	0.95	0.95	0.30	0.00	0.00	0.05	0.05	0.00
66	0.90	1.00	0.30	0.90	0.30	0.00	0.00	0.00	0.00	0.00
67	0.90	1.00	1.00	1.00	1.00	0.00	0.00	0.00	0.00	0.00
68	0.60	1.00	0.70	0.60	0.50	0.00	0.00	0.10	0.10	0.00
69	0.95	1.00	0.15	0.95	0.15	0.00	0.00	0.00	0.00	0.00
70	0.95	0.95	0.95	0.90	1.00	0.00	0.00	0.05	0.05	0.00
71	0.60	1.00	0.60	0.70	0.20	0.00	0.00	0.05	0.05	0.00
72	0.20	0.80	1.00	0.90	0.60	0.00	0.10	0.00	0.00	0.00
73	0.20	1.00	0.70	0.40	0.40	0.00	0.00	0.20	0.20	0.20
74	0.50	1.00	0.90	0.60	0.10	0.00	0.00	0.00	0.00	0.60
75	0.20	0.95	0.85	0.70	0.70	0.05	0.00	0.05	0.05	0.15
76	0.25	0.95	0.85	0.65	0.20	0.00	0.05	0.05	0.05	0.25
77	0.25	1.00	0.70	0.75	0.70	0.05	0.00	0.05	0.05	0.10
78	0.50	0.90	0.90	0.60	0.40	0.00	0.00	0.00	0.00	0.30
79	0.60	1.00	0.90	0.40	0.40	0.00	0.00	0.00	0.00	0.40
80	0.80	0.90	0.90	1.00	0.50	0.00	0.00	0.00	0.00	0.40
81	0.70	0.90	0.85	0.80	0.10	0.00	0.05	0.05	0.05	0.30
82	0.55	0.95	0.75	0.50	0.15	0.00	0.00	0.00	0.00	0.20
83	0.65	0.95	0.85	0.85	0.20	0.00	0.00	0.00	0.00	0.30

Continued on next page

Table A.2: Comprehension Metricss of all games

	CR1	CR2	CR3M	CR3m	CR4	MR1	MR2	MR3M	MR3m	MR4
84	0.40	1.00	0.70	0.70	0.80	0.00	0.00	0.00	0.00	0.20
85	0.20	0.90	0.60	0.90	0.40	0.00	0.00	0.10	0.10	0.10
86	0.10	1.00	0.90	0.80	0.70	0.00	0.00	0.00	0.00	0.10
87	0.50	0.95	0.80	0.85	0.45	0.10	0.00	0.10	0.10	0.00
88	0.30	1.00	0.80	0.65	0.10	0.05	0.00	0.10	0.10	0.10
89	0.30	1.00	0.75	0.70	0.50	0.00	0.00	0.05	0.05	0.05
90	0.90	0.90	0.70	0.70	0.40	0.00	0.00	0.00	0.00	0.00
91	0.80	1.00	0.80	0.40	0.50	0.00	0.00	0.10	0.10	0.10
92	0.70	0.90	1.00	0.80	0.30	0.00	0.00	0.00	0.00	0.20
93	0.80	0.95	0.75	0.50	0.25	0.00	0.00	0.15	0.15	0.10
94	0.95	0.90	0.85	0.40	0.70	0.00	0.00	0.10	0.10	0.10
95	0.85	0.95	0.85	0.60	0.30	0.00	0.00	0.05	0.05	0.20
96	0.60	1.00	0.90	0.70	0.50	0.00	0.00	0.10	0.10	0.10
97	0.70	1.00	1.00	0.10	0.30	0.00	0.00	0.00	0.00	0.10
98	0.90	0.90	0.70	0.40	0.50	0.00	0.00	0.00	0.00	0.10
99	0.95	1.00	0.80	0.75	0.25	0.00	0.00	0.05	0.05	0.25
100	0.90	0.95	0.85	0.45	0.40	0.00	0.00	0.05	0.05	0.25
101	0.70	0.95	0.80	0.55	0.45	0.00	0.00	0.05	0.05	0.15
102	0.80	1.00	0.90	0.80	0.80	0.00	0.00	0.00	0.00	0.00
103	0.70	0.90	0.90	0.60	0.70	0.00	0.10	0.00	0.00	0.00
104	0.80	0.90	0.70	0.60	0.80	0.00	0.00	0.10	0.10	0.00
105	0.85	1.00	0.90	0.70	0.55	0.00	0.00	0.10	0.10	0.20
106	0.80	1.00	0.90	0.35	0.75	0.00	0.00	0.00	0.00	0.10
107	0.95	1.00	0.95	0.80	0.85	0.00	0.00	0.00	0.00	0.00
108	0.90	1.00	0.70	0.70	0.40	0.00	0.00	0.00	0.00	0.00

Continued on next page

Table A.2: Comprehension Metricss of all games

	CR1	CR2	CR3M	CR3m	CR4	MR1	MR2	MR3M	MR3m	MR4
109	0.50	0.90	1.00	0.30	0.20	0.00	0.00	0.00	0.00	0.40
110	0.80	1.00	0.70	0.80	0.20	0.00	0.00	0.00	0.00	0.30
111	0.75	1.00	0.75	0.75	0.25	0.00	0.00	0.10	0.10	0.25
112	0.85	0.95	0.85	0.60	0.25	0.00	0.00	0.00	0.00	0.15
113	0.85	0.90	0.75	0.60	0.30	0.00	0.00	0.15	0.15	0.20
114	0.60	0.80	1.00	0.80	0.30	0.00	0.10	0.00	0.00	0.20
115	0.60	1.00	0.90	0.50	0.40	0.00	0.00	0.00	0.00	0.10
116	0.30	0.90	0.90	1.00	0.40	0.00	0.00	0.00	0.00	0.40
117	0.60	1.00	0.85	0.65	0.15	0.05	0.00	0.15	0.15	0.45
118	0.65	0.95	0.85	0.55	0.20	0.00	0.00	0.00	0.00	0.30
119	0.55	1.00	0.75	0.95	0.20	0.00	0.00	0.00	0.00	0.30
120	0.80	0.90	0.80	0.90	0.80	0.00	0.00	0.00	0.00	0.00
121	0.80	0.80	1.00	0.50	0.50	0.00	0.00	0.00	0.00	0.20
122	0.80	1.00	0.80	0.50	0.50	0.00	0.00	0.00	0.00	0.40
123	0.85	0.95	0.90	0.75	0.45	0.00	0.00	0.00	0.00	0.05
124	0.85	1.00	0.90	0.35	0.60	0.00	0.00	0.05	0.05	0.15
125	0.75	1.00	0.70	0.70	0.55	0.00	0.00	0.10	0.10	0.10
126	0.20	1.00	0.90	0.70	0.10	0.10	0.00	0.10	0.10	0.00
127	0.20	1.00	1.00	1.00	0.20	0.10	0.00	0.00	0.00	0.00
128	0.50	1.00	0.90	0.90	0.50	0.00	0.00	0.10	0.10	0.00
129	0.55	1.00	1.00	0.90	0.25	0.00	0.00	0.00	0.00	0.00
130	0.20	1.00	1.00	0.85	0.10	0.00	0.00	0.00	0.00	0.00
131	0.55	1.00	1.00	0.95	0.25	0.05	0.00	0.00	0.00	0.00
132	0.90	1.00	0.80	0.30	0.20	0.00	0.00	0.00	0.00	0.30
133	0.90	1.00	0.90	0.60	0.40	0.00	0.00	0.00	0.00	0.00

Continued on next page

Table A.2: Comprehension Metricss of all games

	CR1	CR2	CR3M	CR3m	CR4	MR1	MR2	MR3M	MR3m	MR4
134	0.90	1.00	0.90	0.40	0.30	0.00	0.00	0.10	0.10	0.00
135	0.95	1.00	0.95	0.75	0.10	0.00	0.00	0.00	0.00	0.00
136	0.95	0.95	0.90	0.60	0.10	0.00	0.00	0.00	0.00	0.00
137	0.90	1.00	0.95	0.20	0.10	0.05	0.00	0.00	0.00	0.00
138	0.60	1.00	1.00	1.00	0.10	0.00	0.00	0.00	0.00	0.10
139	0.50	1.00	0.80	0.80	0.30	0.00	0.00	0.00	0.00	0.00
140	0.70	1.00	0.90	0.90	0.30	0.00	0.00	0.10	0.10	0.00
141	0.80	1.00	0.95	0.75	0.15	0.00	0.00	0.05	0.05	0.00
142	0.75	1.00	0.85	0.85	0.15	0.00	0.00	0.10	0.10	0.00
143	0.70	1.00	0.95	0.95	0.35	0.10	0.00	0.05	0.05	0.00
144	0.50	0.90	1.00	0.80	0.50	0.00	0.00	0.00	0.00	0.10
145	0.20	0.80	1.00	0.90	0.80	0.00	0.00	0.00	0.00	0.00
146	0.70	0.70	0.90	0.90	1.00	0.00	0.00	0.10	0.10	0.00
147	0.70	0.25	1.00	0.75	0.45	0.00	0.00	0.00	0.00	0.05
148	0.55	0.35	0.95	0.50	0.35	0.00	0.00	0.05	0.05	0.00
149	0.85	0.65	0.95	1.00	1.00	0.00	0.00	0.00	0.00	0.00
150	0.20	0.20	1.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00
151	0.00	0.40	1.00	0.90	0.90	0.00	0.00	0.00	0.00	0.00
152	0.10	0.20	1.00	0.00	0.10	0.00	0.00	0.00	0.00	0.00
153	0.00	0.05	1.00	1.00	0.30	0.00	0.00	0.00	0.00	0.05
154	0.00	0.05	1.00	0.15	0.95	0.00	0.00	0.00	0.00	0.00
155	0.10	0.10	1.00	0.00	0.15	0.00	0.00	0.00	0.00	0.00
156	0.00	0.40	1.00	1.00	0.50	0.00	0.00	0.00	0.00	0.00
157	0.80	1.00	1.00	0.90	0.90	0.00	0.00	0.00	0.00	0.00
158	0.40	1.00	1.00	1.00	0.30	0.00	0.00	0.00	0.00	0.00

Continued on next page

Table A.2: Comprehension Metricss of all games

	CR1	CR2	CR3M	CR3m	CR4	MR1	MR2	MR3M	MR3m	MR4
159	0.00	0.25	1.00	1.00	0.75	0.00	0.00	0.00	0.00	0.00
160	0.85	0.65	0.90	0.65	0.65	0.00	0.00	0.05	0.05	0.05
161	0.50	0.85	1.00	1.00	0.20	0.00	0.00	0.00	0.00	0.00
162	0.80	0.80	0.90	0.50	0.90	0.00	0.00	0.10	0.10	0.10
163	0.90	0.60	1.00	1.00	0.90	0.00	0.00	0.00	0.00	0.00
164	0.80	0.70	0.80	0.40	1.00	0.00	0.00	0.20	0.20	0.00
165	0.80	0.50	1.00	0.70	0.65	0.00	0.00	0.00	0.00	0.00
166	0.90	0.25	1.00	0.80	0.45	0.00	0.00	0.00	0.00	0.00
167	0.80	0.45	0.95	0.50	0.70	0.00	0.00	0.05	0.05	0.00
168	0.80	0.50	1.00	0.90	0.90	0.00	0.00	0.00	0.00	0.00
169	0.90	0.40	1.00	0.10	1.00	0.00	0.00	0.00	0.00	0.00
170	0.90	0.50	1.00	0.90	0.90	0.00	0.00	0.00	0.00	0.00
171	0.90	0.45	0.80	1.00	1.00	0.00	0.00	0.00	0.00	0.00
172	0.95	0.10	0.95	0.00	1.00	0.00	0.00	0.05	0.05	0.00
173	0.95	0.30	0.95	0.40	1.00	0.00	0.00	0.05	0.05	0.00
174	0.90	0.70	1.00	0.60	1.00	0.00	0.00	0.00	0.00	0.00
175	0.60	0.70	1.00	1.00	0.50	0.00	0.00	0.00	0.00	0.00
176	0.60	1.00	0.90	0.80	0.20	0.00	0.00	0.10	0.10	0.00
177	0.95	0.65	1.00	0.90	0.95	0.00	0.00	0.00	0.00	0.00
178	0.30	0.60	0.90	0.85	0.70	0.00	0.00	0.10	0.10	0.00
179	0.80	0.70	0.95	0.85	0.15	0.00	0.00	0.05	0.05	0.05
180	0.40	0.70	1.00	0.50	0.70	0.00	0.00	0.00	0.00	0.00
181	0.20	0.70	0.90	0.80	1.00	0.00	0.00	0.10	0.10	0.00
182	0.50	0.80	1.00	1.00	1.00	0.00	0.00	0.00	0.00	0.00
183	0.75	0.70	1.00	0.80	0.75	0.05	0.00	0.00	0.00	0.05

Continued on next page

Table A.2: Comprehension Metricss of all games

	CR1	CR2	CR3M	CR3m	CR4	MR1	MR2	MR3M	MR3m	MR4
184	0.55	0.35	1.00	0.55	0.25	0.00	0.00	0.00	0.00	0.00
185	0.85	0.55	1.00	1.00	1.00	0.00	0.00	0.00	0.00	0.00
186	0.90	0.70	1.00	1.00	0.10	0.00	0.00	0.00	0.00	0.00
187	0.90	0.60	1.00	0.90	0.20	0.00	0.00	0.00	0.00	0.00
188	0.90	0.50	1.00	0.30	0.00	0.00	0.00	0.00	0.00	0.00
189	0.90	0.15	1.00	1.00	0.15	0.00	0.00	0.00	0.00	0.00
190	0.90	0.10	1.00	0.20	0.10	0.00	0.00	0.00	0.00	0.00
191	0.85	0.50	1.00	0.00	0.15	0.00	0.00	0.00	0.00	0.00
192	0.90	1.00	1.00	1.00	0.20	0.00	0.00	0.00	0.00	0.10
193	0.60	0.90	1.00	0.90	0.90	0.00	0.00	0.00	0.00	0.00
194	0.40	0.80	1.00	0.90	0.10	0.00	0.00	0.00	0.00	0.00
195	0.95	0.60	1.00	0.95	0.05	0.00	0.00	0.00	0.00	0.00
196	0.80	0.75	0.95	0.75	0.70	0.00	0.00	0.05	0.05	0.00
197	0.60	1.00	1.00	0.90	0.30	0.00	0.00	0.00	0.00	0.05
198	0.10	0.30	0.30	0.20	0.50	0.10	0.40	0.40	0.40	0.30
199	0.10	0.40	0.50	0.40	0.40	0.20	0.10	0.20	0.20	0.40
200	0.20	0.50	0.80	0.30	0.20	0.00	0.30	0.20	0.20	0.30
201	0.10	0.50	0.40	0.30	0.45	0.10	0.20	0.20	0.20	0.15
202	0.10	0.50	0.60	0.40	0.25	0.10	0.00	0.20	0.20	0.40
203	0.15	0.55	0.20	0.15	0.10	0.25	0.15	0.35	0.35	0.20
204	0.40	0.40	0.10	0.00	0.10	0.30	0.10	0.50	0.50	0.70
205	0.50	0.60	0.60	0.50	0.20	0.10	0.10	0.00	0.00	0.30
206	0.60	0.30	0.30	0.20	0.50	0.00	0.30	0.50	0.50	0.20
207	0.40	0.60	0.55	0.40	0.45	0.25	0.10	0.10	0.10	0.35
208	0.45	0.60	0.50	0.35	0.20	0.20	0.20	0.25	0.25	0.30

Continued on next page

Table A.2: Comprehension Metricss of all games

	CR1	CR2	CR3M	CR3m	CR4	MR1	MR2	MR3M	MR3m	MR4
209	0.65	0.50	0.75	0.50	0.30	0.15	0.05	0.15	0.15	0.40
210	0.30	0.50	0.50	0.40	0.50	0.30	0.20	0.20	0.20	0.20
211	0.20	0.60	0.60	0.30	0.30	0.10	0.10	0.20	0.20	0.40
212	0.30	0.40	0.60	0.20	0.30	0.10	0.10	0.30	0.30	0.40
213	0.25	0.20	0.50	0.45	0.35	0.20	0.30	0.25	0.25	0.35
214	0.30	0.45	0.60	0.40	0.35	0.10	0.15	0.15	0.15	0.30
215	0.00	0.60	0.25	0.40	0.15	0.05	0.10	0.15	0.15	0.25
216	0.70	0.70	0.40	0.20	0.30	0.00	0.00	0.20	0.20	0.40
217	0.70	0.30	0.60	0.50	0.50	0.00	0.20	0.30	0.30	0.20
218	0.60	0.60	0.30	0.00	0.20	0.10	0.00	0.30	0.30	0.20
219	0.55	0.45	0.45	0.15	0.25	0.10	0.15	0.25	0.25	0.25
220	0.45	0.40	0.65	0.35	0.40	0.05	0.20	0.10	0.10	0.40
221	0.45	0.60	0.40	0.10	0.15	0.05	0.10	0.25	0.25	0.30
222	0.00	0.40	0.30	0.20	0.10	0.30	0.50	0.30	0.30	0.50
223	0.40	0.40	0.80	0.60	0.10	0.10	0.20	0.10	0.10	0.40
224	0.10	0.60	0.20	0.10	0.20	0.00	0.00	0.50	0.50	0.40
225	0.10	0.25	0.55	0.20	0.10	0.10	0.40	0.25	0.25	0.35
226	0.00	0.65	0.40	0.60	0.35	0.15	0.15	0.15	0.15	0.10
227	0.00	0.65	0.45	0.10	0.10	0.05	0.35	0.15	0.15	0.30
228	0.20	0.40	0.60	0.20	0.30	0.00	0.00	0.20	0.20	0.30
229	0.10	0.30	0.40	0.20	0.30	0.20	0.30	0.40	0.40	0.50
230	0.20	0.50	0.50	0.10	0.40	0.10	0.40	0.40	0.40	0.10
231	0.15	0.50	0.60	0.40	0.15	0.15	0.20	0.10	0.10	0.50
232	0.30	0.55	0.55	0.35	0.25	0.15	0.15	0.30	0.30	0.20
233	0.05	0.50	0.30	0.05	0.10	0.20	0.20	0.25	0.25	0.15

Continued on next page

Table A.2: Comprehension Metricss of all games

	CR1	CR2	CR3M	CR3m	CR4	MR1	MR2	MR3M	MR3m	MR4
234	0.30	0.20	0.80	0.30	0.30	0.10	0.40	0.20	0.20	0.20
235	0.30	0.50	0.50	0.20	0.30	0.00	0.10	0.30	0.30	0.40
236	0.40	0.20	0.70	0.20	0.30	0.10	0.30	0.20	0.20	0.40
237	0.30	0.40	0.55	0.30	0.25	0.10	0.30	0.20	0.20	0.40
238	0.45	0.30	0.55	0.35	0.25	0.05	0.25	0.20	0.20	0.25
239	0.60	0.65	0.40	0.20	0.20	0.00	0.05	0.25	0.25	0.35
240	0.60	0.20	0.70	0.60	0.40	0.10	0.10	0.10	0.10	0.40
241	0.50	0.50	0.70	0.30	0.10	0.00	0.00	0.20	0.20	0.60
242	0.70	0.40	0.40	0.40	0.80	0.00	0.20	0.10	0.10	0.10
243	0.65	0.50	0.60	0.35	0.25	0.10	0.15	0.20	0.20	0.45
244	0.65	0.60	0.60	0.15	0.05	0.05	0.05	0.20	0.20	0.35
245	0.70	0.50	0.40	0.45	0.60	0.05	0.05	0.25	0.25	0.30
246	0.20	0.50	0.70	0.60	0.20	0.20	0.00	0.20	0.20	0.30
247	0.60	0.60	0.70	0.70	0.30	0.10	0.10	0.20	0.20	0.20
248	0.10	0.40	0.20	0.10	0.30	0.00	0.10	0.40	0.40	0.10
249	0.15	0.35	0.55	0.45	0.20	0.15	0.20	0.15	0.15	0.45
250	0.35	0.50	0.70	0.35	0.30	0.15	0.20	0.10	0.10	0.30
251	0.30	0.40	0.20	0.30	0.20	0.10	0.20	0.25	0.25	0.25

Table A.3: Behaviour Metrics of all games

	CR	N	TM	Ret	For	Em
0	1.00	1.00	0.00	0.00	0.00	1.00
1	1.00	1.00	0.00	0.00	0.00	1.00
2	1.00	1.00	0.00	0.00	0.00	1.00

Continued on next page

Table A.3: Behaviour Metrics of all games

	CR	N	TM	Ret	For	Em
3	1.00	1.00	0.00	0.00	0.00	1.00
4	1.00	1.00	0.00	0.00	0.00	1.00
5	1.00	1.00	0.00	0.00	0.00	1.00
6	0.30	1.00	0.00	0.75	0.67	0.78
7	0.20	1.00	0.00	1.00	0.50	0.89
8	0.20	1.00	0.00	1.00	0.50	0.89
9	0.20	1.00	0.00	1.00	0.75	0.84
10	0.15	1.00	0.00	1.00	0.67	0.89
11	0.10	1.00	0.00	0.67	0.50	0.95
12	1.00	1.00	0.00	0.00	0.00	1.00
13	1.00	1.00	0.00	0.00	0.00	1.00
14	1.00	1.00	0.00	0.00	0.00	1.00
15	1.00	1.00	0.00	0.00	0.00	1.00
16	1.00	1.00	0.00	0.00	0.00	1.00
17	1.00	1.00	0.00	0.00	0.00	1.00
18	1.00	1.00	0.00	0.00	0.00	1.00
19	1.00	1.00	0.00	0.00	0.00	1.00
20	1.00	1.00	0.00	0.00	0.00	1.00
21	1.00	1.00	0.00	0.00	0.00	1.00
22	1.00	1.00	0.00	0.00	0.00	1.00
23	1.00	1.00	0.00	0.00	0.00	1.00
24	0.00	0.00	1.00	0.00	0.00	0.00
25	0.10	0.00	0.90	0.00	0.00	0.00
26	0.10	0.00	0.90	0.00	0.00	0.00
27	0.05	0.00	0.95	0.00	0.00	0.05

Continued on next page

Table A.3: Behaviour Metrics of all games

	CR	N	TM	Ret	For	Em
28	0.00	0.00	1.00	0.00	0.00	0.00
29	0.00	0.00	1.00	0.00	0.00	0.00
30	0.00	0.00	1.00	0.00	0.00	0.00
31	1.00	1.00	0.00	0.00	0.00	1.00
32	0.30	0.00	0.70	0.00	0.00	0.33
33	0.00	0.00	1.00	0.00	0.00	0.00
34	1.00	1.00	0.00	0.00	0.00	1.00
35	0.50	0.00	0.50	0.00	0.00	0.53
36	0.90	1.00	0.00	0.12	0.89	0.11
37	0.90	1.00	0.00	0.11	0.89	0.11
38	0.90	1.00	0.00	0.12	0.89	0.11
39	0.95	1.00	0.00	0.06	0.95	0.05
40	0.90	1.00	0.00	0.11	0.94	0.11
41	0.95	1.00	0.00	0.06	0.95	0.05
42	0.10	0.00	1.00	0.50	0.50	0.89
43	0.00	0.00	1.00	1.00	0.00	1.00
44	0.00	0.00	1.00	1.00	0.00	1.00
45	0.05	0.00	1.00	1.00	0.50	0.95
46	0.00	0.00	1.00	1.00	0.00	1.00
47	0.00	0.00	1.00	1.00	0.00	1.00
48	0.00	0.00	1.00	1.00	0.00	1.00
49	1.00	1.00	0.00	0.00	0.90	0.00
50	0.30	0.00	1.00	0.75	0.75	0.67
51	0.00	0.00	1.00	1.00	0.00	1.00
52	1.00	1.00	0.00	0.00	0.95	0.00

Continued on next page

Table A.3: Behaviour Metrics of all games

	CR	N	TM	Ret	For	Em
53	0.35	0.00	1.00	0.57	0.88	0.63
54	1.00	1.00	0.00	0.00	0.00	1.00
55	1.00	1.00	0.00	0.00	0.00	1.00
56	1.00	1.00	0.00	0.00	0.00	1.00
57	1.00	1.00	0.00	0.00	0.00	1.00
58	1.00	1.00	0.00	0.00	0.00	1.00
59	1.00	1.00	0.00	0.00	0.00	1.00
60	0.00	0.00	1.00	0.00	0.00	0.89
61	0.00	0.00	1.00	0.00	0.00	0.89
62	0.00	0.00	1.00	0.00	0.00	0.89
63	0.00	0.00	1.00	0.00	0.00	0.95
64	0.00	0.00	1.00	0.00	0.00	0.95
65	0.05	0.00	1.00	0.00	0.33	0.84
66	0.00	0.00	1.00	0.00	0.00	0.89
67	1.00	1.00	0.00	0.00	0.00	1.00
68	0.50	0.00	0.50	0.00	0.75	0.56
69	0.00	0.00	1.00	0.00	0.00	0.95
70	1.00	1.00	0.00	0.00	0.00	1.00
71	0.65	0.00	0.31	0.00	1.00	0.58
72	0.10	0.00	0.90	0.00	0.00	0.11
73	0.20	0.00	0.80	0.00	0.00	0.11
74	0.20	0.00	0.80	0.00	0.00	0.22
75	0.00	0.00	1.00	0.00	0.00	0.00
76	0.05	0.00	0.95	0.00	0.00	0.05
77	0.05	0.00	0.95	0.00	0.00	0.05

Continued on next page

Table A.3: Behaviour Metrics of all games

	CR	N	TM	Ret	For	Em
78	0.70	0.00	0.30	0.00	0.00	0.78
79	1.00	1.00	0.00	0.00	0.00	1.00
80	0.80	0.00	0.20	0.00	0.00	0.78
81	0.80	0.00	0.20	0.00	0.00	0.79
82	0.85	0.00	0.15	0.00	0.00	0.84
83	0.75	0.00	0.25	0.00	0.00	0.79
84	0.20	0.00	0.80	0.00	0.00	0.22
85	0.50	0.00	0.50	0.00	0.00	0.56
86	0.30	0.00	0.70	0.00	0.00	0.33
87	0.50	0.00	0.50	0.00	0.00	0.47
88	0.35	0.00	0.65	0.00	0.00	0.37
89	0.05	0.00	0.95	0.00	0.00	0.05
90	0.10	1.00	0.00	1.00	0.00	1.00
91	0.00	0.00	1.00	1.00	0.00	1.00
92	0.10	0.00	1.00	0.50	0.50	0.89
93	0.05	1.00	0.00	1.00	0.00	1.00
94	0.00	0.00	1.00	1.00	0.00	1.00
95	0.00	0.00	1.00	1.00	0.00	1.00
96	0.50	1.00	0.00	0.33	0.80	0.56
97	0.30	1.00	0.00	0.50	0.67	0.78
98	0.50	1.00	0.00	0.40	0.80	0.56
99	0.45	1.00	0.00	0.60	0.89	0.58
100	0.30	1.00	0.00	0.86	0.83	0.74
101	0.50	0.00	1.00	0.56	0.91	0.47
102	0.20	1.00	0.00	0.67	0.50	0.89

Continued on next page

Table A.3: Behaviour Metrics of all games

	CR	N	TM	Ret	For	Em
103	0.50	1.00	0.00	0.33	0.80	0.56
104	0.00	0.00	1.00	1.00	0.00	1.00
105	0.05	0.00	1.00	1.00	0.50	0.95
106	0.00	0.00	1.00	1.00	0.00	1.00
107	0.05	0.00	1.00	1.00	0.50	0.95
108	0.00	0.00	1.00	0.00	0.00	0.89
109	0.00	0.00	1.00	0.00	0.00	0.89
110	0.00	0.00	1.00	0.00	0.00	0.89
111	0.15	0.00	1.00	0.00	0.50	0.68
112	0.05	0.00	1.00	0.00	0.50	0.89
113	0.00	0.00	1.00	0.00	0.00	0.95
114	1.00	1.00	0.00	0.00	0.00	1.00
115	0.90	0.00	0.10	0.00	0.00	0.89
116	0.90	0.00	0.11	0.00	1.00	0.78
117	0.55	0.00	0.55	0.00	0.89	0.37
118	0.90	0.00	0.05	0.00	0.50	0.95
119	0.45	0.00	0.45	0.00	0.67	0.58
120	0.00	0.00	1.00	0.00	0.00	0.89
121	0.10	0.00	1.00	0.00	0.33	0.67
122	0.00	0.00	1.00	0.00	0.00	0.89
123	0.00	0.00	1.00	0.00	0.00	0.95
124	0.00	0.00	1.00	0.00	0.00	0.95
125	0.00	0.00	1.00	0.00	0.00	0.95
126	0.60	0.00	0.40	0.00	0.00	0.67
127	0.20	0.00	0.80	0.00	0.00	0.11

Continued on next page

Table A.3: Behaviour Metrics of all games

	CR	N	TM	Ret	For	Em
128	0.40	0.00	0.60	0.00	0.00	0.33
129	0.70	0.00	0.30	0.00	0.00	0.68
130	0.20	0.00	0.80	0.00	0.00	0.16
131	0.30	0.00	0.70	0.00	0.00	0.26
132	0.50	0.00	1.00	0.25	0.83	0.44
133	0.20	1.00	0.00	1.00	0.50	0.89
134	0.20	1.00	0.00	1.00	0.50	0.89
135	0.15	1.00	0.00	1.00	0.67	0.89
136	0.15	1.00	0.00	1.00	0.67	0.89
137	0.25	1.00	0.00	0.60	0.80	0.79
138	0.30	0.00	0.40	0.00	0.33	0.78
139	0.30	0.00	0.80	0.00	0.50	0.44
140	0.40	0.00	0.50	0.00	0.25	0.56
141	0.15	0.00	0.60	0.00	0.50	0.79
142	0.25	0.00	0.83	0.00	0.71	0.58
143	0.25	0.00	0.43	0.00	0.25	0.79
144	0.90	0.00	0.10	0.00	0.00	0.89
145	0.90	0.00	0.10	0.00	0.00	0.89
146	1.00	1.00	0.00	0.00	0.00	1.00
147	0.80	0.00	0.20	0.00	0.00	0.79
148	0.85	0.00	0.15	0.00	0.00	0.84
149	1.00	1.00	0.00	0.00	0.00	1.00
150	0.00	0.00	1.00	0.00	0.00	0.00
151	0.00	0.00	1.00	0.00	0.00	0.00
152	0.00	0.00	1.00	0.00	0.00	0.00

Continued on next page

Table A.3: Behaviour Metrics of all games

	CR	N	TM	Ret	For	Em
153	0.00	0.00	1.00	0.00	0.00	0.00
154	0.00	0.00	1.00	0.00	0.00	0.00
155	0.00	0.00	1.00	0.00	0.00	0.00
156	0.00	0.00	1.00	0.00	0.00	0.00
157	0.90	0.00	0.10	0.00	0.00	0.89
158	0.60	0.00	0.40	0.00	0.00	0.56
159	0.00	0.00	1.00	0.00	0.00	0.00
160	0.95	0.00	0.05	0.00	0.00	0.95
161	0.55	0.00	0.45	0.00	0.00	0.58
162	1.00	1.00	0.00	0.00	0.90	0.00
163	1.00	1.00	0.00	0.00	0.90	0.00
164	1.00	1.00	0.00	0.00	0.90	0.00
165	0.85	1.00	0.00	0.18	0.94	0.16
166	0.70	1.00	0.00	0.29	0.93	0.32
167	0.95	1.00	0.00	0.06	0.95	0.05
168	0.00	0.00	1.00	1.00	0.00	1.00
169	0.00	0.00	1.00	1.00	0.00	1.00
170	0.00	0.00	1.00	1.00	0.00	1.00
171	0.00	0.00	1.00	1.00	0.00	1.00
172	0.00	0.00	1.00	1.00	0.00	1.00
173	0.00	0.00	1.00	1.00	0.00	1.00
174	0.00	0.00	1.00	1.00	0.00	1.00
175	1.00	1.00	0.00	0.00	0.90	0.00
176	0.30	1.00	0.00	0.75	0.67	0.78
177	0.00	0.00	1.00	1.00	0.00	1.00

Continued on next page

Table A.3: Behaviour Metrics of all games

	CR	N	TM	Ret	For	Em
178	0.95	1.00	0.00	0.05	0.95	0.05
179	0.60	1.00	0.00	0.50	0.92	0.42
180	0.90	0.00	0.10	0.00	0.00	0.89
181	0.90	0.00	0.10	0.00	0.00	0.89
182	1.00	1.00	0.00	0.00	0.00	1.00
183	0.85	0.00	0.12	0.00	1.00	0.79
184	0.95	0.00	0.05	0.00	1.00	0.89
185	1.00	1.00	0.00	0.00	0.00	1.00
186	0.00	0.00	1.00	0.00	0.00	0.89
187	0.00	0.00	1.00	0.00	0.00	0.89
188	0.00	0.00	1.00	0.00	0.00	0.89
189	0.00	0.00	1.00	0.00	0.00	0.95
190	0.00	0.00	1.00	0.00	0.00	0.95
191	0.00	0.00	1.00	0.00	0.00	0.95
192	0.00	0.00	1.00	0.00	0.00	0.89
193	0.90	0.00	0.10	0.00	0.00	0.89
194	0.40	0.00	0.75	0.00	0.60	0.44
195	0.00	0.00	1.00	0.00	0.00	0.95
196	0.90	0.00	0.11	0.00	1.00	0.79
197	0.35	0.00	0.57	0.00	0.80	0.58
198	0.20	0.00	0.80	0.00	0.00	0.22
199	0.20	0.00	0.80	0.00	0.00	0.22
200	0.20	0.00	0.80	0.00	0.00	0.22
201	0.15	0.00	0.85	0.00	0.00	0.16
202	0.15	0.00	0.85	0.00	0.00	0.16

Continued on next page

Table A.3: Behaviour Metrics of all games

	CR	N	TM	Ret	For	Em
203	0.10	0.00	0.90	0.00	0.00	0.11
204	0.90	0.00	0.10	0.00	0.00	0.89
205	0.90	0.00	0.10	0.00	0.00	0.89
206	0.80	0.00	0.20	0.00	0.00	0.89
207	1.00	1.00	0.00	0.00	0.00	1.00
208	0.90	0.00	0.10	0.00	0.00	0.89
209	0.90	0.00	0.10	0.00	0.00	0.89
210	0.60	0.00	0.40	0.00	0.00	0.67
211	0.50	0.00	0.50	0.00	0.00	0.44
212	0.70	0.00	0.30	0.00	0.00	0.67
213	0.55	0.00	0.45	0.00	0.00	0.58
214	0.20	0.00	0.80	0.00	0.00	0.21
215	0.50	0.00	0.50	0.00	0.00	0.53
216	0.00	0.00	1.00	1.00	0.00	1.00
217	0.10	1.00	0.00	1.00	0.00	1.00
218	0.10	0.00	1.00	1.00	0.50	0.89
219	0.20	0.00	1.00	0.50	0.80	0.79
220	0.20	0.00	1.00	0.60	0.80	0.79
221	0.25	1.00	0.00	0.83	0.80	0.79
222	0.60	1.00	0.00	0.50	0.83	0.44
223	0.90	1.00	0.00	0.12	0.89	0.11
224	1.00	1.00	0.00	0.00	0.90	0.00
225	0.80	1.00	0.00	0.27	0.94	0.21
226	0.90	1.00	0.00	0.12	0.94	0.11
227	0.85	1.00	0.00	0.12	0.94	0.16

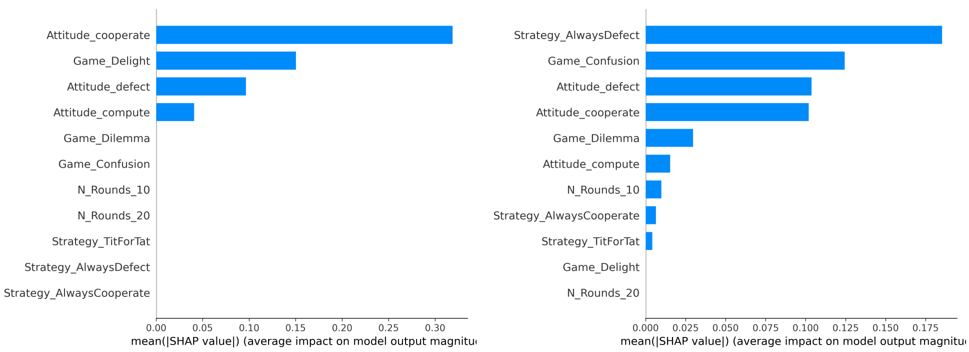
Continued on next page

Table A.3: Behaviour Metrics of all games

	CR	N	TM	Ret	For	Em
228	0.40	1.00	0.00	0.80	0.75	0.67
229	0.40	0.00	1.00	0.50	0.80	0.56
230	0.50	0.00	1.00	0.50	0.83	0.44
231	0.65	1.00	0.00	0.36	0.92	0.37
232	0.70	0.00	1.00	0.23	0.93	0.26
233	0.85	0.00	1.00	0.12	0.94	0.11
234	0.10	0.00	1.00	0.00	0.33	0.67
235	0.20	0.00	0.75	0.00	0.33	0.56
236	0.10	0.00	1.00	0.00	0.33	0.67
237	0.20	0.00	0.67	0.00	0.57	0.74
238	0.10	0.00	0.75	0.00	0.25	0.84
239	0.15	0.00	1.00	0.00	0.60	0.63
240	1.00	1.00	0.00	0.00	0.00	1.00
241	0.90	0.00	0.11	0.00	1.00	0.78
242	1.00	1.00	0.00	0.00	0.00	1.00
243	0.95	0.00	0.05	0.00	1.00	0.89
244	0.95	0.00	0.05	0.00	1.00	0.89
245	1.00	1.00	0.00	0.00	0.00	1.00
246	0.60	0.00	0.38	0.00	0.33	0.67
247	0.70	0.00	0.29	0.00	1.00	0.67
248	0.50	0.00	0.60	0.00	1.00	0.44
249	0.70	0.00	0.27	0.00	0.83	0.68
250	0.65	0.00	0.54	0.00	1.00	0.32
251	0.70	0.00	0.40	0.00	1.00	0.47

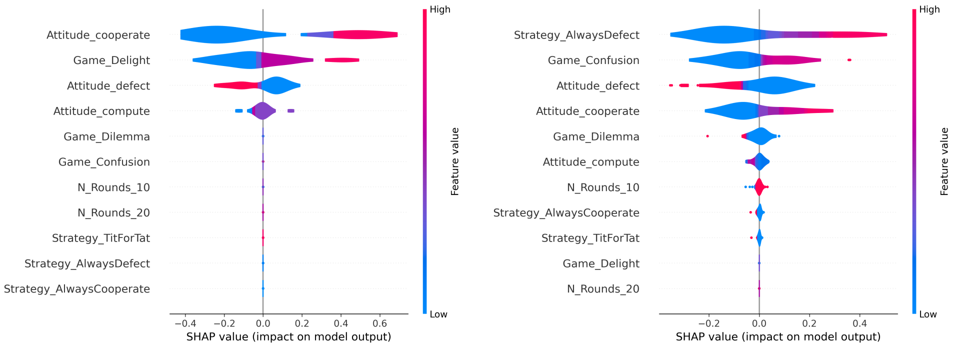
Appendix B

Tool-Based Plots



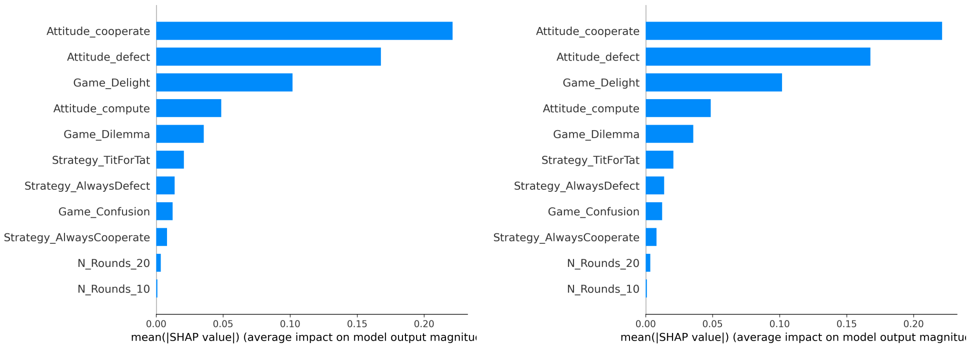
(a) GPT (b) Ministral

Figure B.1: SHAP Feature Importance across models for Niceness.



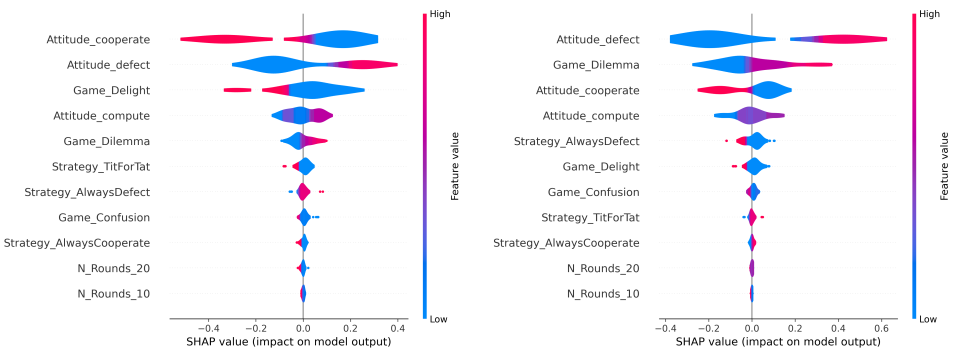
(a) GPT (b) Ministral

Figure B.2: SHAP Violin Plots across models for Niceness.



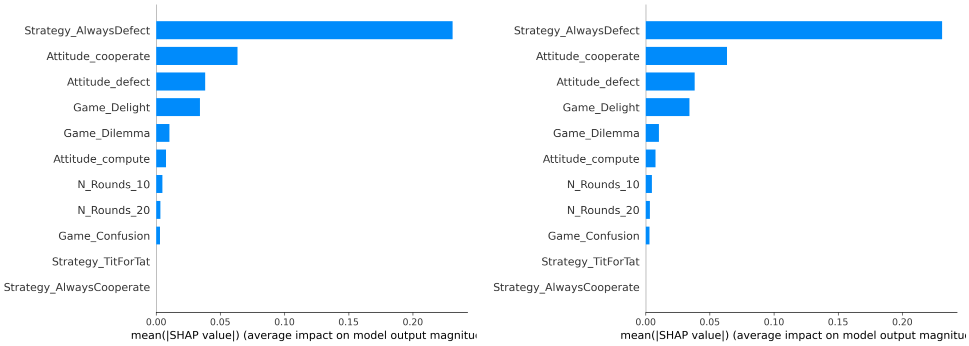
(a) GPT (b) Ministral

Figure B.3: SHAP Feature Importance across models for Troublemaking.



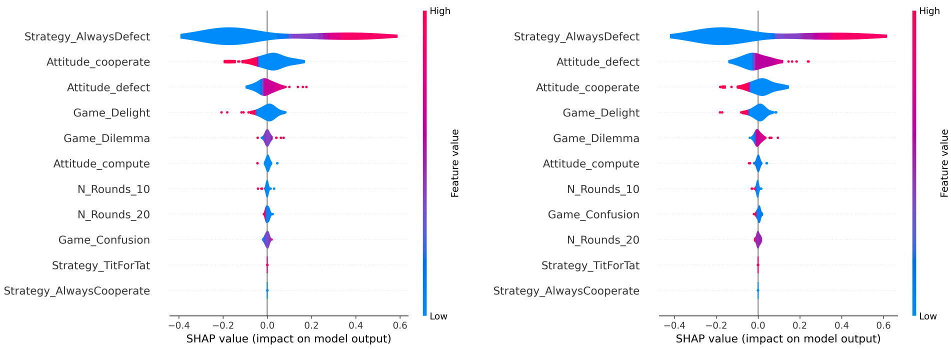
(a) GPT (b) Ministral

Figure B.4: SHAP Violin Plots across models for Troublemaking.



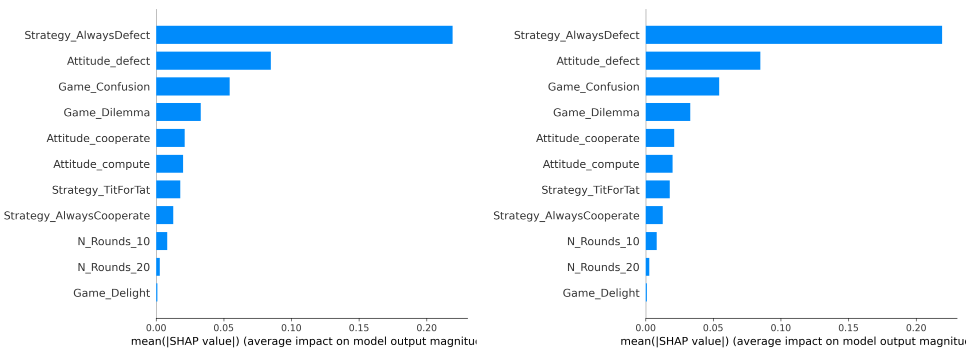
(a) GPT (b) Ministral

Figure B.5: SHAP Feature Importance across models for Retaliation.



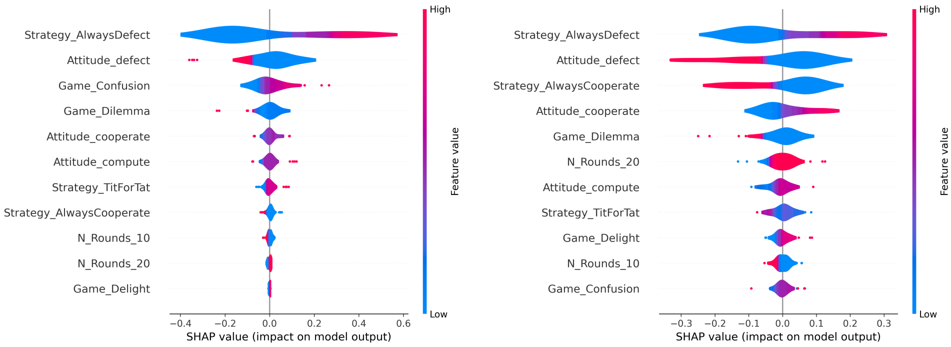
(a) GPT (b) Ministral

Figure B.6: SHAP Violin Plots across models for Retaliation.



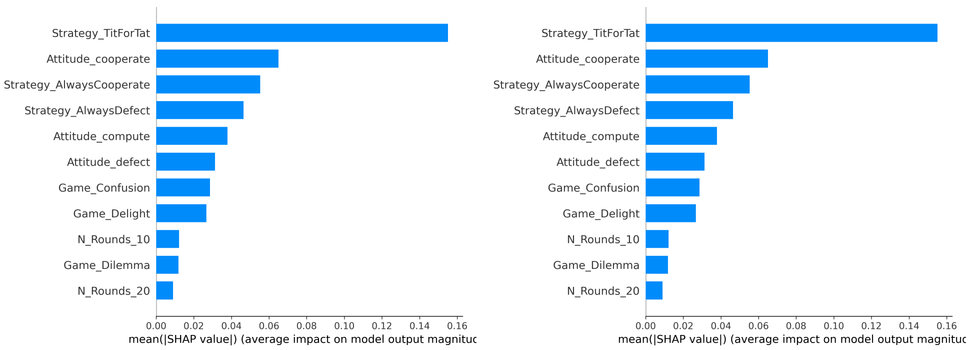
(a) GPT (b) Ministral

Figure B.7: SHAP Feature Importance across models for Forgiveness.



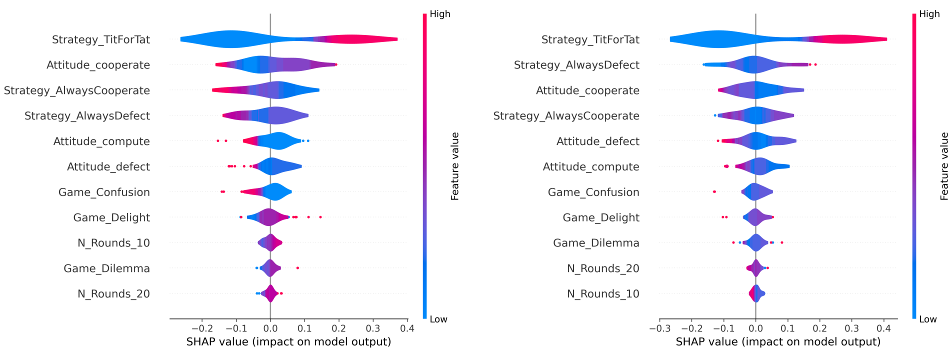
(a) GPT (b) Ministral

Figure B.8: SHAP Violin Plots across models for Forgiveness.



(a) GPT (b) Ministral

Figure B.9: SHAP Feature Importance across models for Emulation.

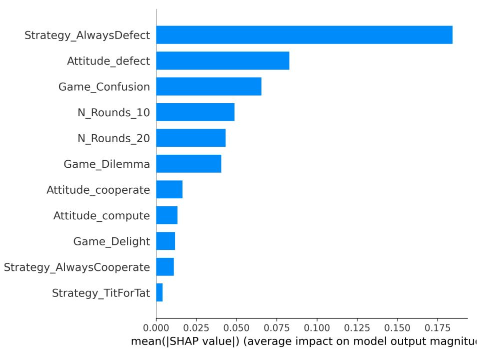


(a) GPT (b) Ministral

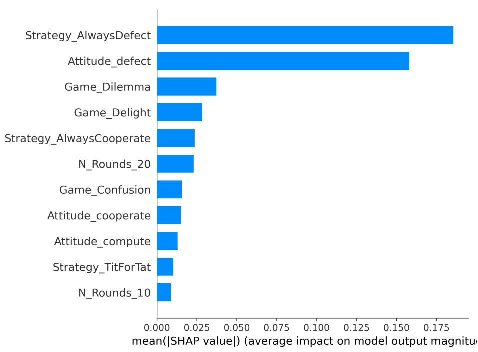
Figure B.10: SHAP Violin Plots across models for Emulation.

Appendix C

Debate-Based Plots

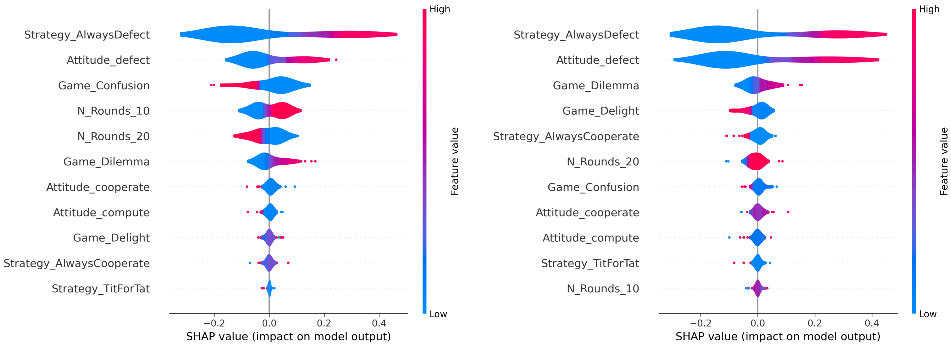


(a) GPT



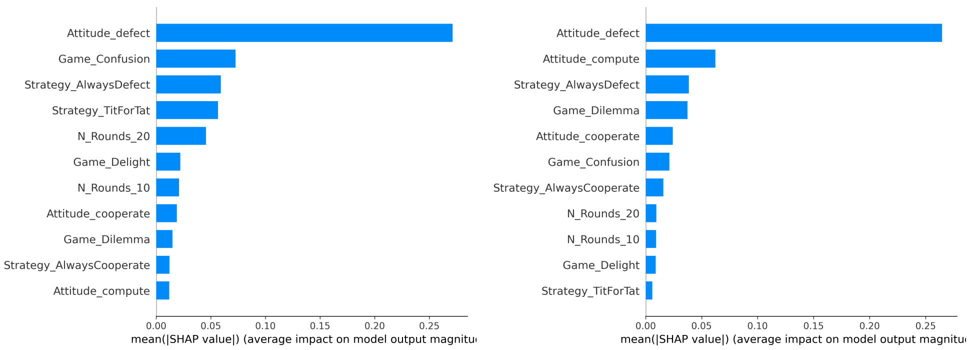
(b) Ministral

Figure C.1: SHAP Feature Importance across models for Niceness.



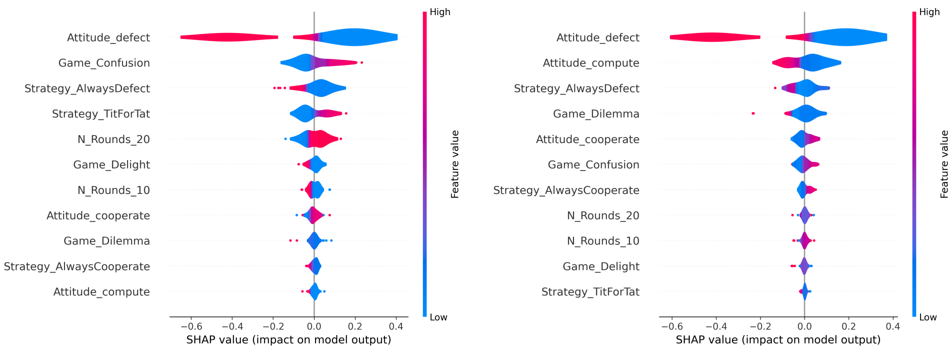
(a) GPT (b) Ministral

Figure C.2: SHAP Violin Plots across models for Niceness.



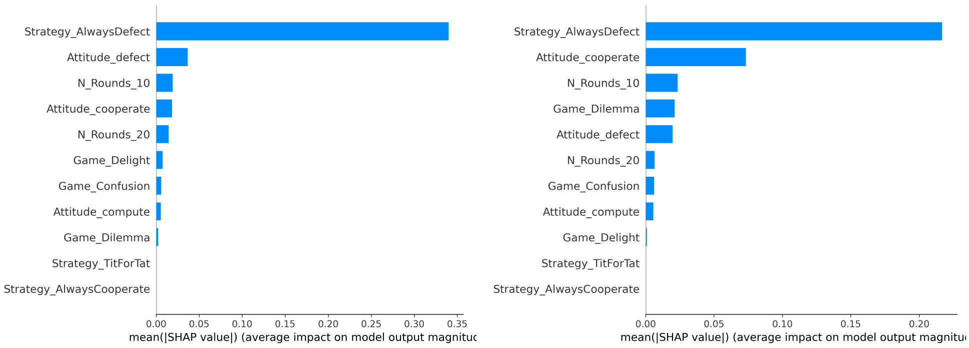
(a) GPT (b) Ministral

Figure C.3: SHAP Feature Importance across models for Troublemaking.



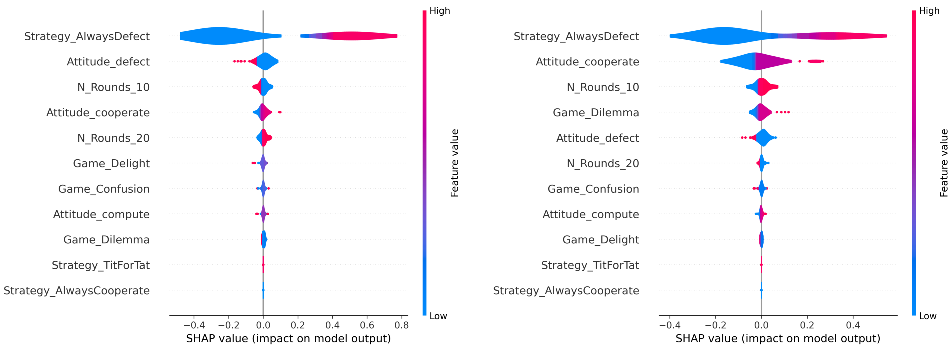
(a) GPT (b) Ministral

Figure C.4: SHAP Violin Plots across models for Troublemaking.



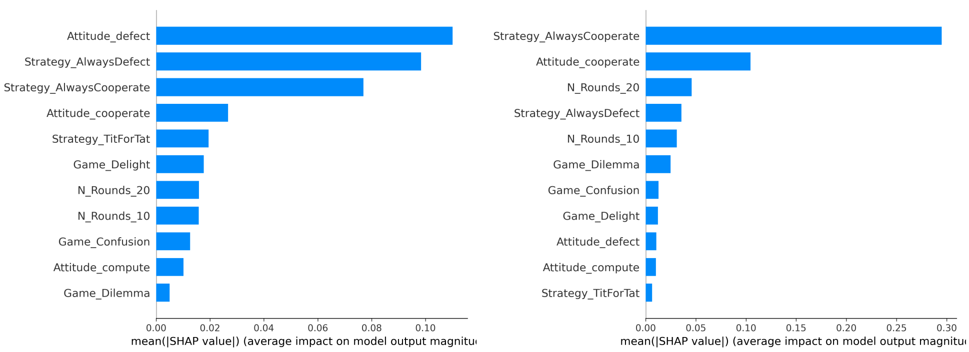
(a) GPT (b) Ministral

Figure C.5: SHAP Feature Importance across models for Retaliation.



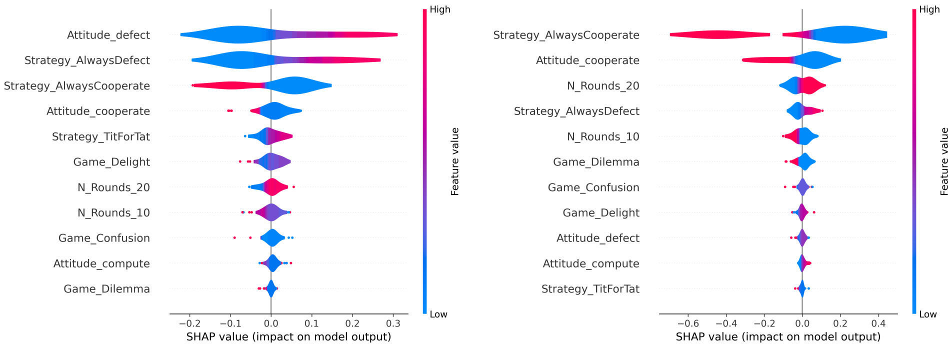
(a) GPT (b) Ministral

Figure C.6: SHAP Violin Plots across models for Retaliation.



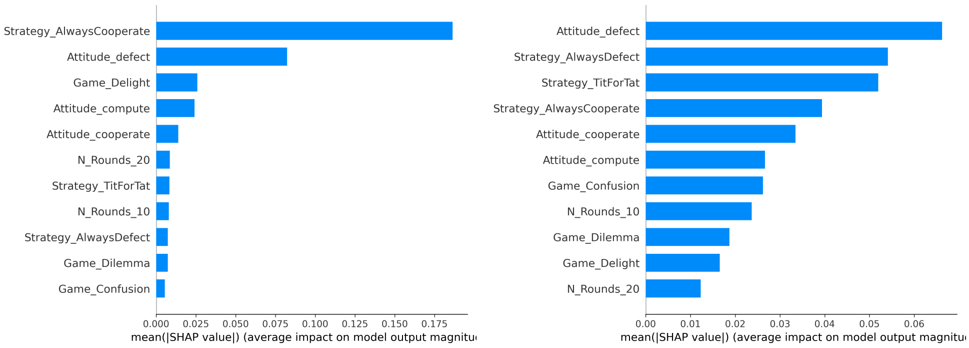
(a) GPT (b) Ministral

Figure C.7: SHAP Feature Importance across models for Forgiveness.



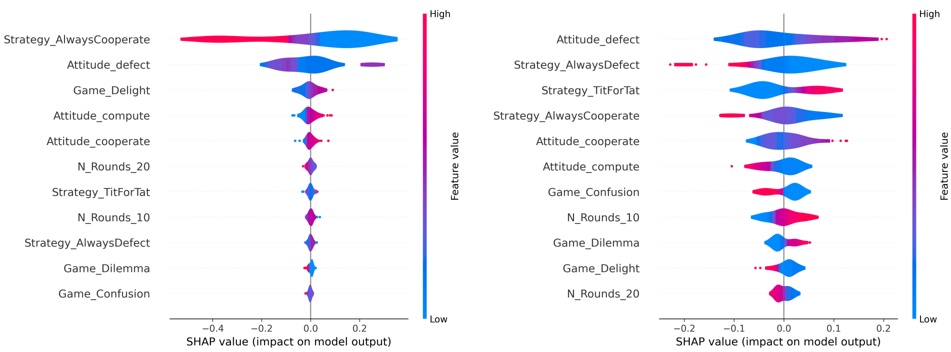
(a) GPT (b) Ministral

Figure C.8: SHAP Violin Plots across models for Forgiveness.



(a) GPT (b) Ministral

Figure C.9: SHAP Feature Importance across models for Emulation.



(a) GPT (b) Ministral

Figure C.10: SHAP Violin Plots across models for Emulation.