ALBERT-LUDWIGS-
UNIVERSITÄT FREIBURG

Faculty of Engineering
Department of Computer Science
Cognitive Computation Lab
apl. Prof. Dr. Dr. Marco Ragni

Predictive Computing -
Predicting Human Behaviour in Games

Master Thesis

Mathias Zink

19th March 2018

Writing period
06. 10. 2017 – 19. 03. 2018

1st Examiner
apl. Prof. Dr. Dr. Marco Ragni

2nd Examiner
Prof. Dr. Bernhard Nebel

Author
Mathias Zink
Schweppermannstraße 78
90408 Nürnberg

Course of study:
Computer Science

Matriculation number:
3107268

Eigenständigkeitserklärung

Hiermit erkläre ich, dass ich diese Abschlussarbeit mit dem Titel

Predictive Computing -
Predicting Human Behaviour in Games

selbständig verfasst habe, keine anderen als die angegebenen Quellen/Hilfsmittel verwendet habe und alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen wurden, als solche kenntlich gemacht habe. Darüber hinaus erkläre ich, dass diese Abschlussarbeit nicht, auch nicht auszugsweise, bereits für eine andere Prüfung angefertigt wurde.

Mathias Zink

Nürnberg, 13. März 2018

Abstract

Menschliches Verhalten in Spielen ist vorhersehbar. Dies soll durch die gezielte Analyse von großen Datenmengen gezeigt werden. Dazu wurden eigens experimentelle Studien durchgeführt, in denen etwa 200 Teilnehmer das klassische Spiel Schere-Stein-Papier gegen verschiedene probalistische bots gespielt haben. So wurden etwa 28000 Spielrunden, von der jede die für das Spiel charakteristische Entscheidung zwischen den vorhandenen Aktionen widerspiegelt, aufgezeichnet. Zusätzlich haben einige der Testpersonen Fragebögen zu ihrem Verhalten beantwortet oder einfache Kognitionsaufgaben abgeschlossen. Die verschiedenen Daten wurden daraufhin nach Mustern und Besonderheiten untersucht. Dabei wurden signifikante Abweichungen von rationalem Verhalten bei den Spielern festgestellt. Features, wie die unausgeglichene Wahl der Aktionen oder das verzerrte Rotationsverhalten nach Sieg oder Niederlage machen das menschliche Verhalten vorhersehbar. Dies soll durch die Interpreatation der Daten durch verschiedene maschinelle Lernmethoden gezeigt werden. Dazu wurde die Vorgehensweise von kollaborativen Filteransätzen, State Vector Machines und Neuronaler Netze verglichen. Als Bewertungsmaß für die genannten Methoden wird die Genauigkeit benutzt, mit der diese die nächste Handlung der Benutzer korrekt vorhersagen können. Außerdem wurden die Informationen, die den verschiedenen Methoden zugänglich gemacht werden, variiert. Dabei handelt es sich beispielsweise um die gewählten Aktionen eines oder beider Spieler über eine variable Anzahl von vergangenen Runden. In die neuronalen Netze wurden außerdem verschiedene Zusatzinputs eingegeben, um zu untersuchen, ob diese zu signifikanten Abweichungen in der Vorhersagegenauigkeit führen. Da diese Vorhersagegenauigkeit abhängig von der Aussagekraft der Merkmale in den Daten ist, wird es möglich, sowohl die Leistung der verschiedenen Methoden als auch die Besonderheiten menschlichen Verhaltens zu interpretieren.

Abstract

Human behaviour in games is predictable. To demonstrate this large amounts of data from simple-decision-making scenarios were analysed and examined for patterns and peculiarities. For this purpose we conducted our own experimental studies, in which over 200 people have played the classic game of Rock-Paper-Scissor (RPS) for about 28000 rounds, against various probabilistic bots. Some of the participants additionally answered questionnaires on their behaviour or completed simple cognitive tasks. Evaluating the data we were able to detect significant deviations from rational behaviour within our test persons. Features such as the unbalanced choice of actions or the predictable cycle behaviour after victory or defeat allow machine learning methods to create meaningful models of the human behaviour during a RPS game. The interpretations of the data are compared by means of collaborative filtering approaches, state vector machines and neural networks in order to draw conclusions about human behaviour. The accuracy with which they correctly predict the next action of the users is used as a measure for the evaluation of these methods. In addition, the information made available to the different techniques has been varied, like the actions selected by one or both players over a variable number of previous rounds. Furthermore, different supplementary inputs are fed into the neural networks to investigate whether these result in significant differences in prediction accuracy. Since this accuracy is dependent on the expressiveness of the characteristics in the data, it is possible to interpret the different methods as well as the peculiarities in human behaviour.

Contents

List of Figures

List of Tables

List of Abbrevations

AI Artificial Intelligence

ANN Artificial Neural Network

BGT Behavioural Game Theory

BR Best Response

C Clockwise [cycle]

CC Counter Clockwise [cycle]

CRT Cognitive Reflection Task

D Draw [result]

FNN Feed Forward Neural Network

GRU Gated Recurrent Unit [network]

GT Game Theory

L Lose [result]

LSTM Long Short Term Memory [network]

NE Nash Equilibrium

ReLU Rectified Linear Unit

RNN Recurrent Neural Network

RPS Rock-Paper-Scissors

STD Standard Deviation

S Stay [cycle]

SVM Support Vector Machine

W Win [result]

1 Introduction

Most people think that artificial intelligence (AI) is a trend that emerged only a few years ago. While the term itself was actually proposed more than 60 years ago, when John McCarthy and some colleagues embossed the term for one of their automata studies in 1956 [31]. They tried to proceed on the conjecture, that any aspect of learning or any other feature of intelligence can in principle be described so precisely that a machine should be able to simulate it. After early success of AI systems on simple problems the progress slowed down in the seventies, when the complexity of the problem space, especially considering the issue of context, surpassed the available computational power. In the years to come AI underwent multiple up's and down's with high hopes and promises which later could not be lived up to. In interplay of various aspects though, AI is now on an upswing for several years starting in the late eighties, as multiple subproblems got solved building upon each other: the computational power grew exponentially, the second spring of neural networks led to fast progress in the field of machine learning and the rise of the technology standard, particularly the permanent connectivity through the internet allowed for a mutual research goal. So it is more common nowadays to build on existing theories and models instead of proposing new ones, meaning that more people advance the same topics. The internet also played a decisive part in the last factor to be mentioned: big data. Through the availability of very large data sets (almost all of them collected and shared through the internet) AI systems and especially the former mentioned machine learning methods are able to learn behaviour from many thousands of examples.

The interplay of all these factors led to the mentioned trend of AI, being one of the most discussed topics of today's society. To extend our knowledge and understanding of the broad field of artificial Intelligence we have to plunge deeper

into specific regions. To do so we can first of all divide AI into four strongly connected categories: Thinking Rationally, Acting Rationally, Thinking Humanly & Acting Humanly [45, p. 1ff]. With the first two being mostly based on the laws of logic and the task of acting accordingly, they try to find the best, most rational, solution to a problem and try to determine what is needed to carry this out. Acting Humanly can easiest be understood with the Turing test, proposed by Alan Turing in 1950, where we try to create intelligence that can not be distinguished from human intelligence anymore. The last remaining category deals with the matter of thinking humanly, which will be professed a little more in the next paragraph, as it is the most important one for this thesis. It is needless to say, that none of the four can be mastered without comparable progress in the others.

Thinking humanly: the cognitive modelling approach. To say that a given program thinks like a human, we first of all have to determine how humans think. This can be done through introspection (catching your own thoughts as they go by), psychological experiments (observing another person in action) or through brain imaging (observing a brain in action). If a program's input-output behaviour matches corresponding human behaviour, that is evidence that some of the program's mechanisms could also be operating in humans. (cf. [45])

One of the first to try this were Newell and Simon in 1961. With their 'General problem solver' they compared the reasoning steps of their program with those of human subjects [34]. From this point the interdisciplinary field of *cognitive science* emerged, bringing together computer models from AI and experimental techniques from psychology to construct precise and testable theories of the human mind.

In this work we choose an experimental study to obtain information about human behaviour. Particularly we use the well known and studied game of Rock-Paper-Scissors. The game is probably one of the oldest ever invented by mankind, with written evidence from China going back over 2000 years [33]. It also counts as one of the fairest mechanisms to resolve conflicts, as it was just recently used to settle an argument in an auction house [50]. Due to its simplicity and the perfect balancing we are able to transfer the players decisions almost exclusively onto his cognition and his perception of his opponent, making it an interesting game to study. Furthermore its specifics reoccur in samples and areas all over the world, with the most prominent examples in nature and economics [6, 26, 28, 42]. For example, can the cyclic behaviour be physically observed, when looking at three species of side-blotched lizards, whose male population periodically alternates exactly like the game predicts [46].

1.1. Contribution

In this thesis we try to combine a classical and simple cognitive experiment with modern machine learning to obtain a better understanding of human cognition. A self acquired big data collection allows to find and predict patterns in human decision making behaviour. Multiple different methods from machine learning are compared to find out which technique predicts the human way of thinking, for this specific problem, the closest and also which features are the most important to look out for.

1.2. Organization of the Thesis

We start the analysis by comparing already existing work to Rock-Paper-Scissors psychology and computational methods. Chapter 3 describes the theory of the game and the different machine learning methods. Afterwards we examine the characteristics of our experiment and present the data in section 4. This data is then used to evaluate and interpret the different models in chapter 5. Lastly we conclude the results and give a prospect on possible improvements for the future.

2 Related Work

Earlier work in this topic can be divided into two sections. Behavioural studies of humans in decision making scenarios, especially in games and particularly in Rock-Paper-Scissors (RPS). And secondly into work, that is trying to model such problems with computational methods.

2.1. Rock-Paper-Scissors in Psychology

From a psychological point of view the most influential survey of Rock-Paper-Scissors was conducted by Wang, Xu, and Zhou [52] in 2014. The three researchers from China were able to show a significant cycling behaviour in their experimental subjects depending on the players success or failure in the previous round of the game, which even improved when increasing the pay-off and therefore the value for the player. They showed that winning a round increases the probability of the subjects to use the same throw again, while losing makes them more likely to swap. This corresponds to the well known win-stay-lose-shift strategy. An important thing to note here is that the users were matched against a random user after every round. Therefore they only have knowledge about their own last throw and no information about their current opponent whatsoever. This leads to an experiment pointing solely on the human decision process given success or failure. Most other experiments including this thesis, aim on getting information about the decision-making whilst duelling a single opponent repeatedly.

Some parts of this approach were readopted 2016 by Dyson et al. [16]. They claim that negative reinforced cycling has a much bigger impact than the positive one and show that cycle behaviour depends on the used throw as well as the outcome. Additionally they propose that the cycle's direction tends to continue onto the next

turns.

The two sources above as well as multiple others suspect that the distribution at which the three possible actions are chosen, also is not as uniform as it is expected to be. In all of the mentioned surveys 'Rock', is the most chosen one, obtaining a pick rate around 35.5%. Followed by 'Paper' and 'Scissors', with the latter one being mostly the least played.

2.2. Computational RPS

Since many years people try to master simple and lately even not so simple games by using machines to analyse the game's theory and therefore facilitate the decision-making process for the human player. It is important to note that almost all computational work on this subject is concerned with creating a machine that beats his human opponents as convincingly as possible, while the goal of our work is to predict human behaviour and draw conclusions on his cognition from it. The fact that a correct prediction of the opponents action corresponds to a won round in the game, allows us to still compare the results with reservation.

2.2.1. Early approaches

One of the first appearances of a RPS playing machine (that was not solely probalistic) was in the year 2000, with Ali et al. playing the game with a genetic algorithm [3]. They specifically used an algorithm that stores the frequency their opponent used a particular throw after matching his last three choices before that. Which means they tried to match their opponents history-string of length three. As a defence mechanism they used a random caprice, which probability increases with the number of successive losses. The algorithm achieved a winrate around 53.5% (35.3% including ties) versus human players (10 players with 30 rounds).

2.2.2. Machine learning approaches

A more recent approach was presented by Pozzato et al., using a machine learning algorithm based on Gaussian mixture models, where again the last three throws get analysed. In contrast to the last approach though, this time both the opponents and the robots turns were captured to train the algorithm. The authors achieved a winrate of up to 36.6% on a dataset of 650 games from multiple users.

Big data. Two notable approaches prior to this one have tried to solve the RPS problem via big data collection. Firstly the New York Times published an interactive online game in 2011 that falls back onto a database of over 200000 rounds of play with an undisclosed winning percentage. The idea and the initial database can be attributed to Shawn Bayern, whose website [7] runs the game already since 2001. After four rounds of random play it starts matching the exact history (own and opposite throws), with its database and uses the strategy that beats the throw used by most users in the same situation. Essentially, it treats the game as a forth-order Markov process, predicting your action, depending on the average action of past players. Both websites are still running up to this date but the underlying database for the better promoted NYTimes site does not seem to have changed since its first publication.

Pomerleau extends this approach using an up to 10-th order Markov chain, allowing also imperfect matches, which get weighted accordingly. This allows the author to reach good results even with a comparably small database of 13000 rounds. Over another 4000 rounds the bot achieved a winrate of 53.05%. More interesting in this case though is the fact that including ties the winrate almost reaches 40%, which means that ties only occur in around 24.5% of games, which is significantly lower than in random play.

3 Fundamentals

3.1. Rock-Paper-Scissors Theory

A formal description and game theoretical analysis of the Rock-Paper-Scissors game family was introduced by Panumate et al. [38] in 2016. They established the format *RPS(n,b,s,r)*, which means that n players simultaneously show a move among b possible moves with possible s winning regulations at each round out of r round matches in total.

In the basic and best known variant *RPS(2,3,1,r)*, the n=2 players can choose from b=3 different actions: R (Rock), P (Paper) and S (Scissors), all depicted by a specific motion of their hand. Depending on cultural background the players will count to three (1-2-3, Ro-sham-bo, Rock-Paper-Scissors, Schnick-Schnack-Schnuck, etc.) while swinging their fist simultaneously and revealing their chosen throw on four. Conditioned on the following rules player one or two receives a point (or none if their chosen throws match): Action 'Rock' beats action 'Scissors', which in turn beats 'Paper', which in turn is better than 'Rock'. This behaviour is visualized in Figure 3.1.

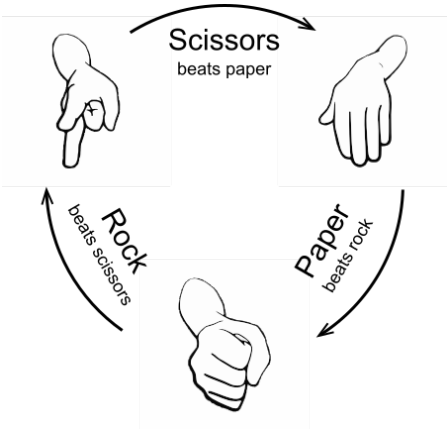


Figure 3.1.: Strategic interaction between the possible actions in a standard game of Rock-Paper-Scissors

It has been shown that from a game refinement point of view our classical model

should be played for r=9 rounds, resulting in a game refinement score [49] similar to extremely balanced and popular board games like Go or Chess and even close to classical sports like foot- or basketball. This is the reason why in the continuation of this thesis we will be using the model RPS(2,3,1,9) unless stated otherwise.

3.2. Decision Theory

Decision theory describes the study of reasoning underlying an agent's choice during a decision-making process, which marks the cognitive process of selecting a logical option among several. When trying to make a good decision, the agent must weight the positives and negatives of each option, and consider all the alternatives. For effective decision making, the decision maker must be able to predict the outcome of each option, and based on all these items, determine which choice is the best for that particular situation. Therefore making a decision in a stochastic environment is a very complex and computational intensive task. Observing humans during a decision-making process gives great insight into the persons values, preferences and beliefs. Comparing those for multiple different people in the same scenario allows us to infer valuable information for cognitive science.

3.2.1. Game theory

Closely related to the field of decision theory, game theory (GT) is mainly concerned with the interactions of agents, whose decisions affect each other. It provides a mathematical framework to represent and analyse rational decision-making and formalizes the outcomes and corresponding utilities for rational agents. This allows for easy reproduceability and interpretation of the decisions. GT is also an important part of many research areas, like economics, political science, computer science, and philosophy, where it is widely used for studying behavior in social settings to describe and predict the types of cooperation, conflict, and coordination observed in groups of human decision makers (cf. [8]).

In game theory we have to distinguish between decisions with perfect and imperfect information, meaning knowing everything about the current state and its possible evolutions or being uncertain about parts of it. This might be induced by simultaneous decision-making, by disclosed information like hand cards, or even by being unsure about the incentive of your counterpart. RPS is a game of imperfect information, as the outcome of every single round depends on the decisions all players make equally and you do not know those decisions beforehand.

Single-move games. Most games consist of multiple moves that are executed sequentially. To break down a game into its components it is best to isolate and analyse single turns. Choosing the action that maximizes your utility in every single situation maximizes your overall pay-off. A single move game can be defined with three components (cf. [45, p. 667f]):

- The *players* or agents, who will make the decisions. In our Rock-Paper-Scissors example we just use a two players game, but it is possible to expand this to any amount of players. Every player is represented by a unique set of capitalized letters A/B or its name Anna/Barney.
- All *actions* the players can choose from. Normally displayed as lower-case names like 'one' or 'playRock'. They are mostly, but not necessarily the same set for all players. For RPS the eligible actions are 'playRock', 'playPaper' and 'playScissors'
- Lastly a *pay-off function* that specifies the utility to each player for each combination of actions by all players. For single move games, this can be represented by a matrix, known as the *strategic* or *normal form* of the game. The pay-off matrix for RPS looks the following:

Table 3.1.: Pay-off matrix for a single round of two players Rock-Paper-Scissors

	B: playRock	B: playPaper	B: playScissors
A: playRock	A=0, B=0	A=-1, B=+1	A=+1, B=-1
A: playPaper	A=+1, B=-1	A=0, B=0	A=-1, B=+1
A: playScissors	A=-1, B=+1	A=+1,B=-1	A=0, B=0

The two players A and B each choose one of the three actions and receive a pay-off of 1, 0 or -1 depending on the opponents pick. This represents the three possible outcomes of a single round of RPS: winning, drawing or losing.

Strategy profile. Every player in a game will adopt a strategy, which means he will choose one of the available actions with probability p. If $p = 1$ for one of the actions and correspondingly $p = 0$ for all others, this strategy is called *pure*. For most games, especially in a multiple-move game, it is advisable to use a *mixed strategy*, which has a probability distribution over all possible actions. Depending on each player's choice of strategy, and therefore its strategy profile, we can determine the game's outcome, a numeric utility value for each player, equalling the pay-off.

For every strategy of a player, there exists a *best response* (BR) for other players, meaning a strategy that yields the most favourable outcome for themselves, given the initial players strategy. In RPS this simply is the throw that beats the other player.

If there exists a strategy profile for a player that is better than all others, this is called the dominant strategy (weakly /strongly). A rational player would always choose a dominant strategy over any other, since this improves its expected utility. If there's a dominant strategy for all players, the combination of those is called a *dominant strategy equilibrium.*

The mathematician John Nash proved that every game has at least one equilibrium. The concept of equilibrium in game theoretic scenarios is therefore being referred to as *Nash equilibrium* (NE) in his honour. An equilibrium is essentially a local optimum, where no player can improve its pay-off by switching strategy, given the other players stick to their current choice. Meaning every player is already playing its best response to the opponent's strategy. Due to its cyclic dominance RPS does not have a pure strategy NE, instead we have to search a mixed strategy, that yields a probability distribution over all possible actions.

In a single turn of RPS the choice of strategy has no impact on the outcome of the game, since the expected utility of every action in the game is constant (zero), independent of your own choosing. This means every participants gain or loss in utility is exactly balanced by the loss or gain of all other participants, which makes it a *zero-sum game*. In a single-move of RPS we do not have any prior knowledge about our opponent, apart from the fact that he's rational, since this is one of the pre-requisites of game theory. If Player A therefore chooses to play 'Rock', he's equally likely to receive a pay-off of 0,-1 and 1. The outcome is purely defined by chance.

Repeated games. When played only for a single move, the outcome of many games is dominated by chance or will end up in a local optimum only [53]. In a repeated game, players face the same decision repeatedly, but each time with knowledge about the history of all players' previous choices. The ability to remember this complete history at all times is called *perfect recall*. A strategy profile for a repeated game, includes the choices for each player, at each time step, for every possible history of previous choices. The payouts of the underlying single-move games are added over time to portray the outcome of the repeated game.

When looking at RPS in a repeated setup we might be able to find an equilibrium. The best strategy in any multi-move two-player zero-sum game is to figure out what your opponent is going to do and choose the action that does best against it, essentially finding the best response. Vice versa your opponent is most likely doing the exact same.

Intuitively selecting a pure strategy does not allow you to react to your opponent's strategy at all, while you make his task as easy as possible. This clearly will not maximize your utility. Furthermore this also can not be an equilibrium, since at any point, where you do not win you would want to switch your strategy, which already contradicts with the definition of an equilibrium. Same holds for mixed strategies including a probability distribution over two of the actions only. Therefore our only chance to find an equilibrium is to find a mixed strategy that alternates between all three throws. The probability distribution over all actions, that will lead us to the mixed strategy equilibrium will be determined in the following.

The pay-off matrices, inferred from Table 3.1 above for A and B, respectively are:

$$A = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix} \qquad B = \begin{pmatrix} 0 & +1 & -1 \\ -1 & 0 & +1 \\ +1 & -1 & 0 \end{pmatrix}$$

Assuming player A wants to use all of his three strategies. Let p_{AR} be the probability of player A choosing action 'playRock', p_{AP} the probability of him choosing action 'playPaper' and p_{AS} the probability of taking 'playScissors', with

$$p_{AS} = 1 - p_{AR} - p_{AP}. \tag{3.1}$$

His strategy profile would therefore look like $\sigma_A = (p_{AR}, p_{AP}, 1 - p_{AR} - p_{AP})$ and the pay-offs for Player B against this strategy are given by:

$$\sigma_A P_B = \begin{pmatrix} p_{AR} \\ p_{AP} \\ 1 - p_{AR} - p_{AP} \end{pmatrix} \begin{pmatrix} 0 & +1 & -1 \\ -1 & 0 & +1 \\ +1 & -1 & 0 \end{pmatrix} = \begin{pmatrix} 2p_{Ap} + p_{AR} - 1 \\ 1 - p_{AP} - 2p_{AR} \\ p_{AR} - p_{AP} \end{pmatrix}$$

Equating the pay-offs mathematically, leaves us with two relevant equations:

$$2p_{Ap} + p_{AR} - 1 = p_{AR} - p_{AP} \quad \Leftrightarrow \quad p_{AP} = \frac{1}{3} \tag{3.2}$$

$$1 - p_{AP} - 2p_{AR} = p_{AR} - p_{AP} \quad \Leftrightarrow \quad p_{AR} = \frac{1}{3} \tag{3.3}$$

Inserting this into Eq. (3.1) , completes the strategy profile for player A:

$$\sigma_A = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$$

Since the game is symmetric we can just exchange the roles and receive $\sigma_B = \sigma_A$.

The result of choosing all the actions with the same probability, meaning choosing them completely random, is the expected outcome. Since both players have the exact same knowledge about previously played rounds and in theory perfect recall about it, any reasoning one player does, can be recreated by his opponent and therefore used to counter it. However this is also known to both players, which makes it an infinite cycle. This behaviour is often referred to as *reasoning dilemma* by advanced RPS players. In our chosen variant of the game the cycle automatically repeats itself after three stages.

With this standing in the way of a meaningful prediction of your opponent's strategy, the only way of losing is if your own strategy is exploitable. So the game theoretic best strategy in RPS is to make yourself unpredictable, marking the **mixed strategy Nash equilibrium** at $[(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}), (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})]$. This correspondingly, should be the strategy used by a rational player, who competes with another rational player.

The fact that humans are only restricted rational, only have restricted recall and additionally are not able to make multiple completly random choices in a row [30, 51], allows us to still predict their behaviour. The field of behavioural game theory, described in the next section, deals with this theory.

3.2.2. Behavioural game theory

The field of behavioural game theory (BGT) arose after multiple researchers noticed deviations from the behaviour the game theorists suggested. This started with the work of Allais [4] in 1953, who discovered that his probands behaviour did not mirror the payoff they receive, but he could not find an explanation yet. Experimental studies from the 70's finally demonstrated that behaviour that occurred in economic markets showed peculiarities, that could not be grasped with the theory, but could be proven experimentally [48]. From this point on researchers examined the conditions that caused this divergence from rational behaviour [19]. The task of behavioural game theory is to describe those unusual phenomena, in contrast to GT, which tries to prescribe a correct reaction given crucial assumptions. BGT allows us to formalize new assumptions using empirical data, like we do in this thesis. Some of the for this work most relevant behavioural deviations are described in the following.

The most important one for this thesis is the inability of humans to create random sequences [30, 51]. It was shown, that random sequences created by humans reliably deviate from random ones. They normally exhibit too few long runs of the same sample, too many alternations, and, consequently, deviation frequencies that are too close to the actual event probabilities (cf. [14]). It has also been shown though that humans are better at creating those sequences if in a game scenario where they belief it is the best strategy to be random [41], which should be the case for the players in our experiments. The second point is that humans are proven to act irrationally, especially following disappointing/negative events [17]. But they mostly still converge to the rational equilibrium in the long run when facing the problem repeatedly, as they adjust their strategies based on past experience [29]. Meaning that even though single decisions might be contradictory to what game theory suggests, humans should ultimately be able to learn from those mistakes and return to a more rational behaviour again. It is important to note that with increasing incentive[12, 47], especially monetary incentive [11, p. 38f], this kind of behaviour is even enhanced.

Mirrored onto our RPS problem, all those should heavily reflect onto the throw distribution and especially on the earlier mentioned cycling behaviour of the probands. With the empirical data we collect in our experiments we are hopefully able to detect those behavioural peculiarities and allow our machine learning methods to predict our probands decisions with more than random accuracy, as proposed by classical game theory. The fundamentals of the used prediction methods will be described in the next sections.

3.3. Machine Learning

The field of machine learning is mainly concerned with finding solutions to problems for which algorithms are difficult to develop [36]. Instead, one uses applied stochastic and lets a model *learn* from data, until it can infer results from previously unknown data as well [21]. A machine learning algorithm can be classified into the categories supervised or unsupervised. It learns either from data only, i.e. unsupervised, or from data and their respective desired results, i.e. supervised. These desired results for an input are also called labels. There exist multiple different machine learning methods, some of them described in the next sections, but firstly we examine the probably most successful one: *Artificial neural networks*.

3.3.1. Artificial neural networks

Artificial neural networks (ANNs) are computational models, inspired by biology, as they are based upon natural neural networks, that form nerve cell networks in the brain [21]. Neuroscience suggests that human cognition is caused by electrochemical activation, which is transmitted by synapses interconnecting these cells. The activation of a single neuron can be defined as the aggregated incoming activation. When a certain activation threshold is reached, it 'fires' and distributes the activation to subsequent cells, thereby stimulating them. The processes that make up human cognition are controlled by these neural networks in the brain, which demonstrate their impressive computing power. Artificial neural networks represent a mathematical abstraction of the biological process that tries to exploit their potential. (cf. [44])

Structure. Figure 3.2 shows a schematic representation of a single artificial neuron with inputs $x_1, ..., x_n$ and output y. An ANN consists of layers of such nodes, forming large network structures. Between the neurons of different layers lie weighted edges, resulting in a directed graph. The first layer is the input into the network, consisting of the so-called features, i.e. the properties of the data we want to learn from. The last layer provides the features or labels of the

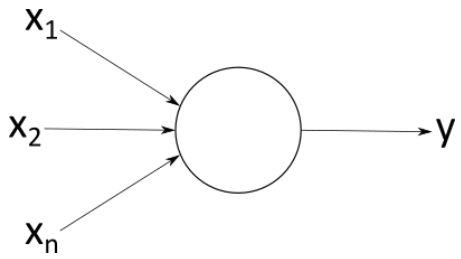


Figure 3.2.: Scheme of a single neuron

network output. Both the individual neurons as well as the entire network thus have inputs and outputs. An exemplary *Feed Forward network* (FNN) with two inputs, a variable number of hidden nodes and layers and three outputs is shown in Figure 3.3. FNNs are the first and simplest form of neural networks, where the information only moves in one direction: forward. Additionally all neurons are connected to all neurons in the previous as well as the next layer.

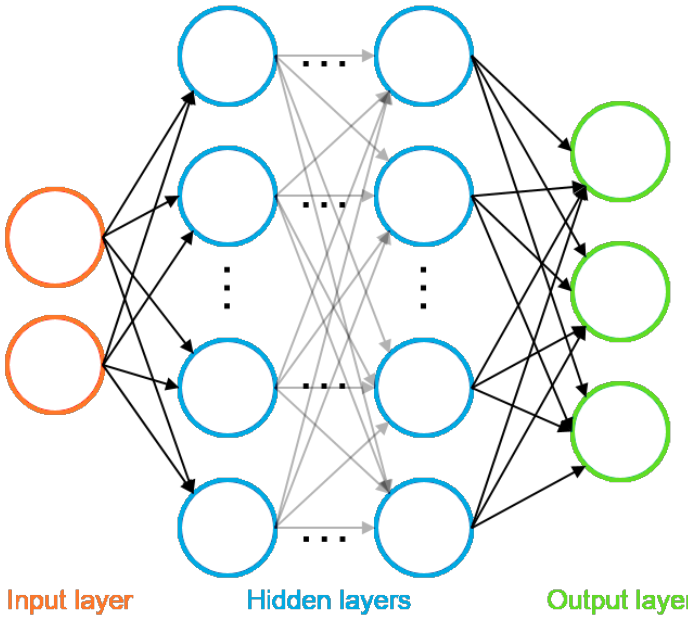


Figure 3.3.: Exemplary structure of a fully connected feed forward neural network.

Learning process. The essential part of artificial networks is, that all connections between two neurons i,j are assigned a weight $w_{i,j}$, defining their influence to the output, whilst all neurons have a bias, which marks the shift of their activation function. The activation of neuron j is aggregated by summing the incoming activations a_i from all n preceding neurons, weighted with $w_{i,j}$ and passing it through the neural activation function σ:

$$a_j = \sigma(x) = \sigma(\sum_{i=1}^{n} w_{i,j} a_i)$$

The activation σ can be set to any function, that transforms the input into the outgoing activation. Table 3.2 lists a selection of the commonly used functions. The activation function has a big impact on the output of each neuron and is therefore a important parameter to determine.

Adapting the weights and biases until they describe the problem shown to the network the best, is the core task of the learning process. To do this, we first have to measure how big the error currently is, i.e., the difference between our output and the desired output for the given input. This is frequently done using methods like the *mean squared error*

$$MSE = \frac{1}{n} \sum_{i} = 1^n (y_i - \bar{y_i}),$$

with y marking the real and $\bar{y}$ the expected output, or the *cross-entropy error*

$$CE = -\sum_{c=1}^{M} \ln(p_{i,c}) \cdot q_{i,c},$$

with M being the number of classes, p the predicted probability that i is of class c and q being a binary indicator that indicates if c is the correct classification for i. Subsequently, the factors on which this error depends, i.e. all weights and biases in our network, are gradually changed in such a way that the error is minimized. This is done using *(Stochastic) Gradient Descent* with different optimizers and *Backpropagation*. Two powerful tools, that allow for more effective and faster calculations in the network and thus contribute decisively to the success of neural networks. However, the exact theories and processes behind the two of them exceed the share of this section and are skipped accordingly. Good sources for both can be found in Nielsen's book on neural networks [36].

Table 3.2.: Exemplary activation functions

Name	Function	Derivative
Linear	$\sigma(x) = x$	$\sigma'(x) = 1$
Hyperbolic tangent (Tanh)	$\sigma(x) = \tanh(x)$	$\sigma'(x) = 1 - \tanh^2$
Logistic (Sigmoid)	$\sigma(x) = \frac{1}{1+e^{-x_j}}$	$\sigma'(x) = \sigma(x) \cdot (1 - \sigma(x))$
Rectified linear (ReLU)	$\sigma(x) = \max(0, x)$	$\sigma'(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases}$

Problems. Unfortunately, artificial neural networks, in addition to their great success in many areas, also cause problems. Mainly is it very difficult for people to understand the problem-solving knowledge of neural networks [25]. Meaning that you mostly do not know how a problem was solved, allowing you to reproduce it with other methods, but only that it has been solved. This also makes it increasingly hard to detect and localize errors in your model.

The most limiting factor of neural networks though, is definitely their need to be learned on very big datasets. It can be mathematically shown that neural networks are able to approximate any function with arbitrary precision given a sufficiently large network and enough data [5]. However, due to a potentially extensive number of parameters, the task of finding an optimal parameter configuration is immensely complex and the existence of a dataset describing the problem sufficiently is also not guaranteed.

A common problem during learning of neural networks occurs when it is *Overfitting* the data. The reason for this lies in too much specialization onto specific details of the training data and the associated lack of ability to generalize to foreign data. This can be indebted to badly selected data, insufficient randomization of the examples or simply training for too long on the same set. To avoid that, neural networks should be validated. For this purpose, some of the data is split away before training starts. This *Validation set* is then used to test the learned fit of your model to new, unseen data.

3.3.2. Recurrent neural networks

A subtype of neural networks are *Recurrent Neural Networks* (RNN's), which, in contrast to feed forward networks, also allows its neurons to use its own output as additional input in the next timestep. This means it allows loops in its structure, making it especially suited to model the relationships of sequences. RNN's are more complex to train, but with possible feedback loops they show greater proximity to the structure of biological brains, as they do not start their 'thinking process' from scratch every time.

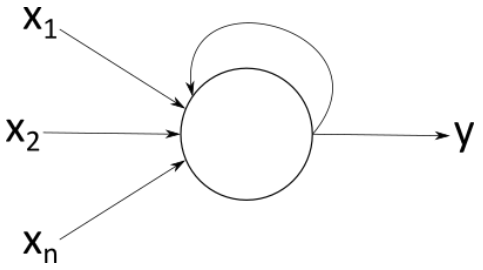


Figure 3.4.: Scheme of a single recurrent neuron

Long-Short-Term-Memory networks

Long-Short-Term-Memory networks (LSTM's) are a special kind of RNN, with improved capabilities to learn long-term dependencies. They were introduced by Hochreiter and Schmidhuber [23] in 1997 and refined and popularized by many in the following years. They have been incredibly successful solving all kinds of sequence problems, like speech recognition, translation tasks or time series prediction. The key difference to a normal RNN is the internal structure, using four internal neural layers, instead of one. With each of them updating the cell state in an individual manner, allowing the network to control its output more precisely and pass on information from earlier cycles. A comprehensible explanation of the different gates and their interplay and any further reading on LSTM's can be found on Christopher Olah's website [37].

Gated-Recurrent-Unit networks

The last network type, we try to predict our data with, is the *Gated-Recurrent-Unit* (GRU). Its structure features three layers, making it slightly less complex and therefore easier and faster to learn than a LSTM. Unfortunately this also means that the information that comes from memory (long term information) is modelled less precisely.

3.3.3. Support vector machines

For comparative purpose we also model the data with other learning methods. For example using a *Support Vector Machine* (SVM), which aims to divide a set of objects into two classes. To do so it tries to fit a hyperplane into the object space, in such a way that as many objects as possible are classified correctly, while as few objects as possible are close to the class boundaries. For multi-class classification the SVM repeats this binary classification until it reaches the given amount of different classes. Similar to the training of neuronal networks, we use a labelled training set to train the classifier or in this case the function of the hyperplane, and a validation set to verify our results afterwards.

In our specific case we use C-SVM classification, which tries to minimize the function:

$$\frac{1}{2}w^T w + C\sum_{i=1}^{N}\zeta_i$$

subject to the constraints

$$y_i(w^T\phi(x_i + b) \geq 1 - \zeta_i \qquad \text{and} \qquad \zeta_i \geq 0$$

where C is the capacity constant, w is the vector of coefficients, b is a constant, and represents parameters for handling inseparable input data. The index i labels the N training cases. ζ represents the class labels and x_i represents the independent variables. It should be noted that the larger the C, the more the error is penalized, thus an ill-considered chosen C can lead to overfitting. (cf. [24])

The hyperplane naturally cannot be 'bent', so that a clean separation with a hyperplane is only possible if the objects are linearly separable. This condition is generally not fulfilled for real training object sets. In the case of non-linearly separable data, Support Vector Machines use the kernel trick to confiscate a non-linear class boundary [2, 9]. The idea behind the kernel trick is to convert the vector space and the training vectors in it into a higher dimensional space. In a room with a sufficiently large number of dimensions - in case of doubt infinite - even the most interleaved vector set becomes linearly separable. In this higher dimensional space the separating hyperplane is now determined. During the back transformation into the lower dimensional space, the linear hyperplane becomes a nonlinear, possibly even unconnected hyperplane, separating the training vectors neatly into the given classes. (cf. [9])

The kernel parameter K is used to transform data from the input to the feature space. There are various kernels that can be used in support vector machine models

[24], making it one of the parameters to optimize:

$$K(x_i, x_j) = \left\{ \begin{array}{ll} x_i^T \cdot x_j & \text{linear} \\ (\gamma x_i^T \cdot x_j + r)^d & \text{Polynomial} \\ \exp(-\gamma |x_i \cdot x_j|) & \text{Radial basis function (RBF)} \\ \tanh(\gamma x_i^T \cdot x_j + r) & \text{Sigmoid} \end{array} \right\}$$

3.3.4. Recommender systems

A *recommender system* is a subclass of an information filtering system, that tries to predict a rating or preference a user would give to a specific item. Systems like this are around us at all times, for example when shopping online or watching a movie. While carrying out those actions the system will analyze your behaviour and try to give you 'useful' hints on what you might like. There exist two major approaches to recommendation systems: collaborative filtering and content-based filtering [20, 43].

The first, and relevant for this thesis, uses large amounts of data to compare a person's behaviour to other users with a similar profile. It is based on the assumptions, that users who agreed in the past will also do so in the future and that they will like similar items now as they did in the past. It can be implemented either item-based, meaning we have an item and try to find other items that are similar to this and infer the rating/output accordingly, or user-based, meaning for a new item, take all similar users and mean/median their opinion. The key advantage of user-based collaborative filtering is, that the content itself is irrelevant and therefore does not have to be interpretable by the machine, making it possible to recommend very complex items like movies without understanding why or what the users like about this item. [10]

Content-based filtering on the other hand tries to recommend items with similar content than what you are currently looking at. This idea can not be transferred onto our prediction problem and will therefore be disregarded in this work.

4 Experiment

4.1. Data Acquisition

4.1.1. Rock-Paper-Scissors-Data

Frontend

To acquire a significantly large dataset we set up a webpage where probands were able to play the game of Rock-Paper-Scissors online. The layout of the website is shown in Figure 4.1. To achieve a comfortable environment for the majority of users a small experiment was conducted beforehand, comparing different looks for header, animations, buttons and the wording of the result texts. The webpage was build using *HTML* and *Css* for appearance and *JQuery* for functionality.

Backend

The backend was implemented using exclusively open source libraries. Whenever a new player registers, a new game is started or even whenever a user plays its next turn a request is sent to the backend. This consists of a webserver running Apache and a Database using *MariaDB's MySQL*. All requests are sent via *REST* (Representational State Transfer), which uses the Url and corresponding parameters to specify its requests. The commands are processed on the server and the answer is sent back to the web interface with *JSON* (JavaScript Object Notation) data format using *AJAX* (Asynchronous JavaScript and XML) calls. Additionally a request is sent to the database, using *Entity Framework*, to register the new user, game or turn. Figure 4.2 visualizes the interaction of the different modules in front- and backend.

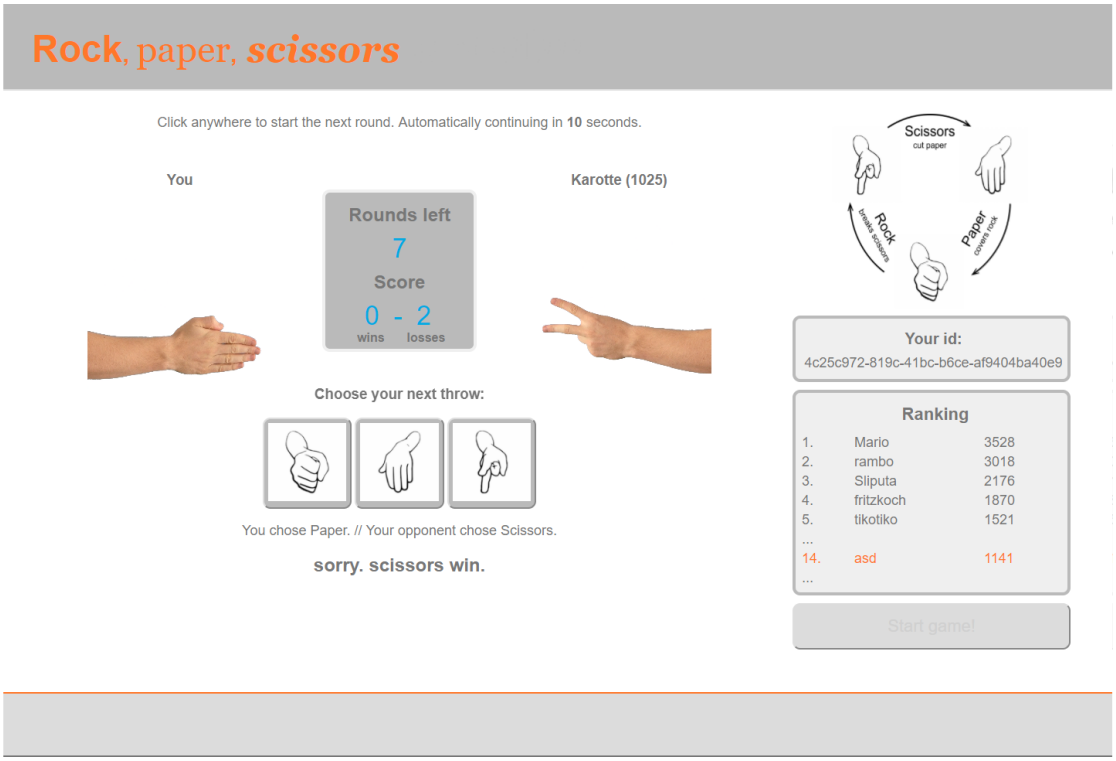


Figure 4.1.: Layout of the experiment-website

Ranking. To keep up the motivation of our probands, we added a ranking system to the website. Every player in-/decreases its own ranking depending on its performance in the last match. The best five players and their rating as well as your own position in the rating and your score is shown on the game screen at all times (Figure 4.1). From the beginning we advertised the experiment with a price pool, with guaranteed prices for the highest ranked players, but also for random participants. With this distribution we hope to attract a broad base of participants, who play a small amount of games for the potential random award and a small group of players competing for the top ranks, leading to a more detailed strategy profile for a selection of players.

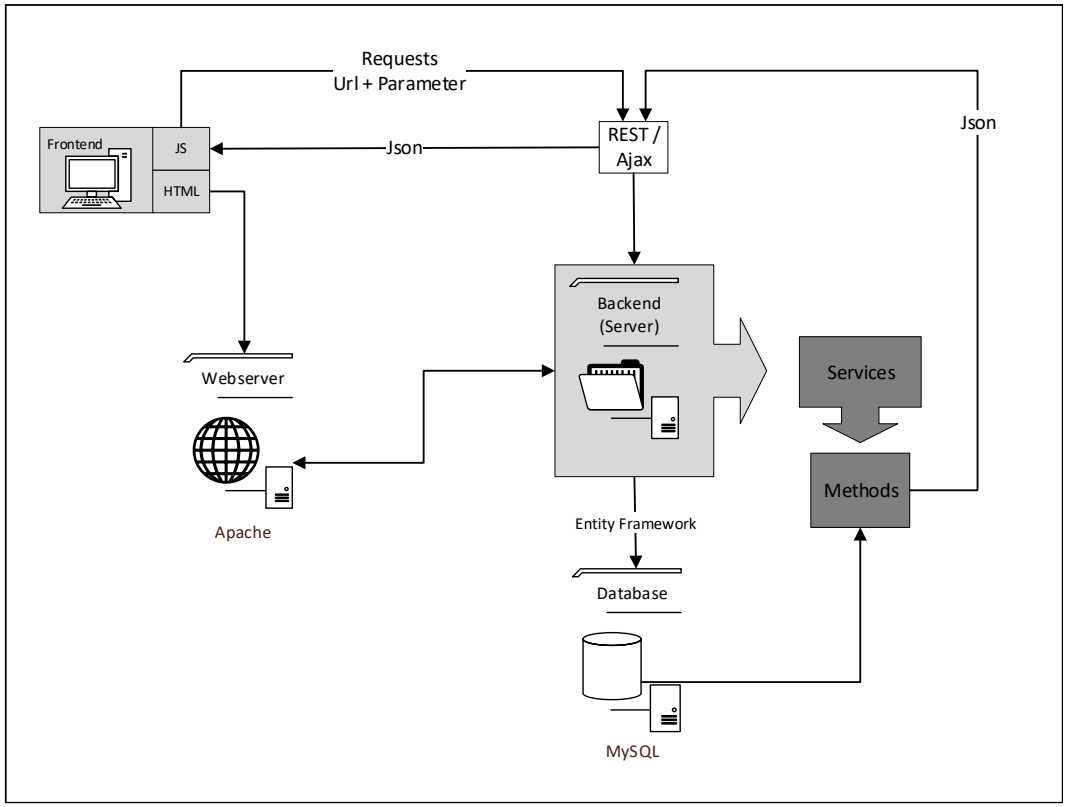


Figure 4.2.: Visualization of the interplay of front- and backend for the experiment's webpage

Bots. We used a total of seven different bots to oppose our human players. The bots get selected randomly with a small constraint on the players given skill bracket.

- RandomBot: Using one of the three actions at complete random
- CounterOwnBot: Using the action that would have won against his last throw in 50% and the others in 25% respectively
- CounterLastBot: Using the action that counters the users last throw in 50% and the others in 25% respectively
- Sequence3/4/5Bot: Uses a random sequence of 3/4/5 throws and repeats them until the game ends
- GameFrequencyBot: Counters the throw that has been used least by the player up to this point

All bots also react to players who repeat a single action multiple times, countering it after three repeated occurrences.

Turing. During data acquisition we tried to give the user the impression of playing against a human opponent, even though he was actually playing versus a bot at all times [54]. To hide this fact we used multiple features that would normally only occur while playing against a human opponent. Firstly whenever starting a new game, the webpage is waiting for multiple seconds to find an opponent, before continuing with the game. During this time we chose a name and the corresponding rating out of the actual ranking list, randomly chosen out of all players with a similar rating. This information is then displayed above the opponents animation (Figure 4.1), indicating that this is the name of the player currently opposing him. Furthermore after every turn there is a chance of a small wait time until the opponent is ready as well, conditioned on your own ready up time. Lastly we used an animation of a human hand for both players, showing the common swinging of the fist during ready up and the known signs for 'Rock', 'Paper' and 'Scissors' after the choices are made. This feature scored highest during the layout questionnaire, asking for the animation, that gives the best impression of playing against a human opponent.

4.1.2. Questionnaires

After every game of RPS we asked the users to fill out a short questionnaire. It includes a rating from 1 to 5 of your enjoyment of the game, the opponents skill level and his rationality, as well as a selection menu for your own and your opponents strategy. The menu included the following options:

- No strategy involved. Just random clicking.
- Intentionally tried to make a random choice at every point.
- Specific pattern decided beforehand. (Like a specific cycle of throws)
- Gambits. (Predefined pattern of 3 throws)
- Read my opponents strategy and used the throw that beats it.
- Mostly use the throw that beat my last throw.
- Mostly use the throw that beat his last throw
- Other. Please Specify below.

On the basis of this questions we try to get a better understanding of the players thinking process during a game, and also hope to find correspondences between the users or games.

4.1.3. Cognition tasks

For the second set of experiments we added two cognition tasks before the actual gameplay to gain a better grasp on the cognitive abilities of our participants.

Cognitive reflection task

We start with a short questionnaire, called the *Cognitive reflection task* (CRT), which was originally proposed by Frederick in 2005. According to him, there are two general types of cognitive activity called 'system 1' and 'system 2'. System 1 is executed quickly without reflection, while system 2 requires conscious thought and effort. The test consists of three questions, shown in Figure 4.3, that each have an obvious but incorrect response given by system 1 . The correct response requires the activation of system 2. For system 2 to be activated, a person must note that their first answer is incorrect, which requires reflection on their own cognition. We use this to evaluate to what extent the cognitive reflection translates into game specific decision-making.

N-Back task

Furthermore we asked the subjects to solve the *single N-back task* [27] up to a level of $n = 3$. In this task the user is presented a sequence of stimuli one-by-one and has to decide respectively if the same stimuli was presented already n steps ago. In our case we used a sequence of letters (visual stimuli), because this transferred best to our initial assignment. We confine the task from $n = 1$ to $n = 3$, since those tend to provide the most expressive results [32] and more tests on this topic would lead to less time and motivation left for our main experiment. With the N-back task we examine the capability of our probands to remember past turns and if this broadcasts into their game behaviour.

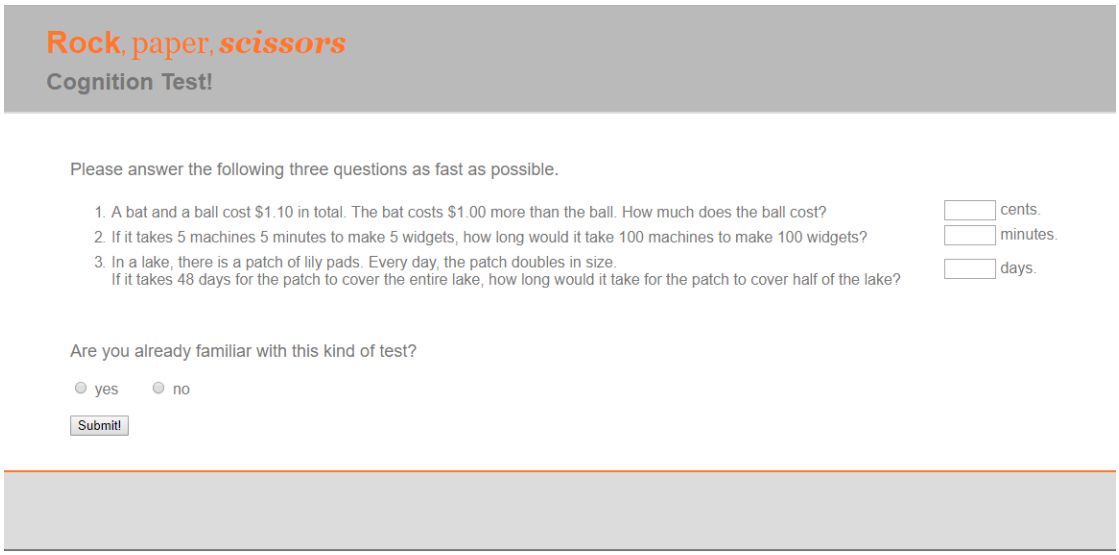


Figure 4.3.: Excerpt of the Cognitive reflection task as posted on the website

4.2. First Experiment

For our first experiment we collected data of 185 different participants. Their demographic stats are visualized in figure 4.4. Basic statistics about the experiment can be found in Table 4.1.

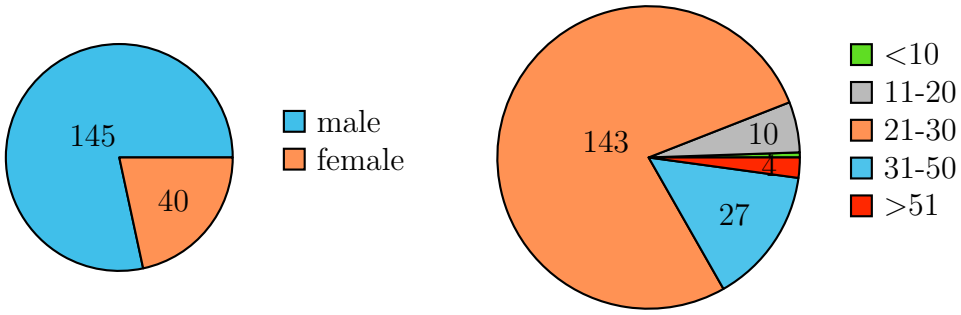


Figure 4.4.: Gender (left) and age (right) distribution of the users participating in the first experiment

Table 4.1.: Descriptive statistics about the first experiment

	Value	STD
Users	185	
Games	2273	
Turns	28841	
Questionnaires	1592	
Average games per user	12.07	43.07
Average games per user (no top5)	5.98	12.80
Average turns per game	13.18	2.65

4.2.1. Throw distribution

Overall throw occurrence

As mentioned in the Related work section, multiple surveys prior to this one suggest that the three different actions do not get selected with even probability. Figure 4.5 shows, that this also is not the case here. As in most other studies action 'Rock' gets selected the most, while 'Scissors' is used the least. The null hypothesis, that this distribution should be uniformly distributed can be rejected with high confidence (Rock: $p < .001$, z-value = 3.49; Paper: $p = 0.4839$, z-value = -0.70; Scissors: $p=0.0048$, z-value = -2.82). The overuse of action 'Rock', as well as the below average usage of 'Scissors' are correspondingly highly significant deviations from rational behaviour. The variance of 'Paper' is not.

First throw

Another claim is that male players tend to start a game with 'Rock', as it is the most aggressive and dominant. As shown in Figure 4.6 this is not the case in our study. Rather the opposite, as the probability for a female user is almost twice as high to start with 'Rock'. Men tend to start with action 'Paper', while '*Scissors*' is, like in the overall throw-distribution, the least chosen. The relatively uniform distribution for the opposing bots, which is listed in the appendix in Table A.1, shows that this behaviour is not evoked by our bot structure. This implies that the deviations from rationality at the first throw of a game is indeed a cognitive decision made by the participants.

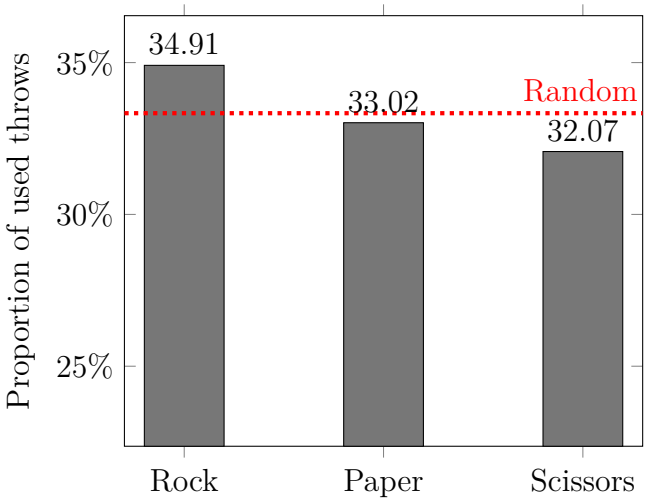


Figure 4.5.: Throw-distribution over all turns and users in the first experiment, expected uniform distribution marked in red

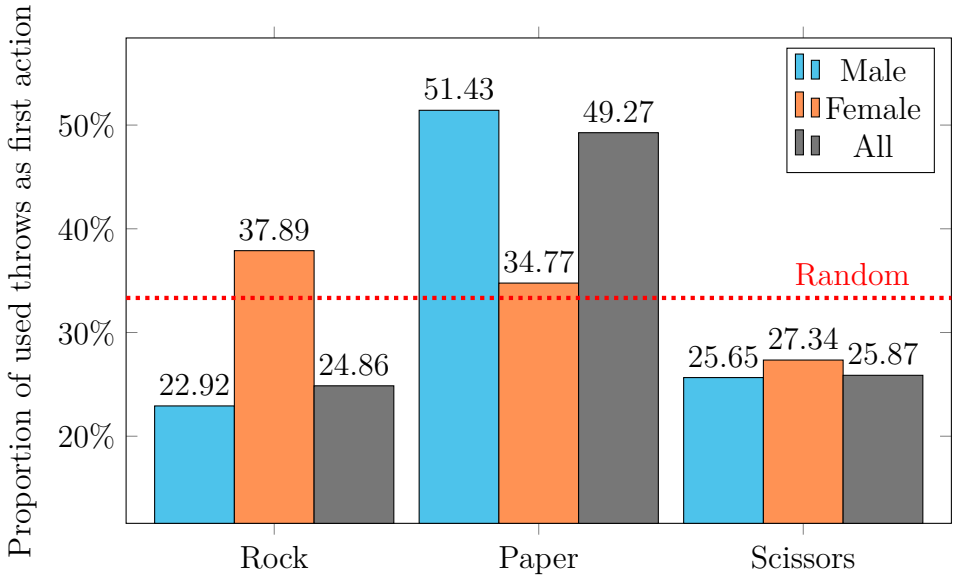


Figure 4.6.: Throw-distribution of single first action in every game, divided by gender, expected uniform distribution marked in red

4.2.2. Cycling behaviour

Additionally we examined the cycling behaviour of our participants. Earlier literature proposes that users tend to continue their behaviour after winning and change strategy after losing. We have to keep in mind though that the underlying experiment was designed in a way, that the users were not matched against each other repeatedly. Figure 4.7 shows how people react during our experiment in a repeated format and it clearly indicates a different behaviour. Users tend to change their actions significantly more often than expected (Stay: $p<.001$, z-value = -16.9), notably even more after winning. This might be caused by the players fear of making themselves predictable by repeating their action and also the mentioned fact that humans tend to change their selection too often when trying to produce random/unpredictable sequences. The second thing that stands out is the well above average proportion of clockwise rotations. As visualized earlier in Figure 3.1 the clockwise rotation is R $\rightarrow$ S $\rightarrow$ P $\rightarrow$ R, meaning that players choose the action that would have lost against their own last action.

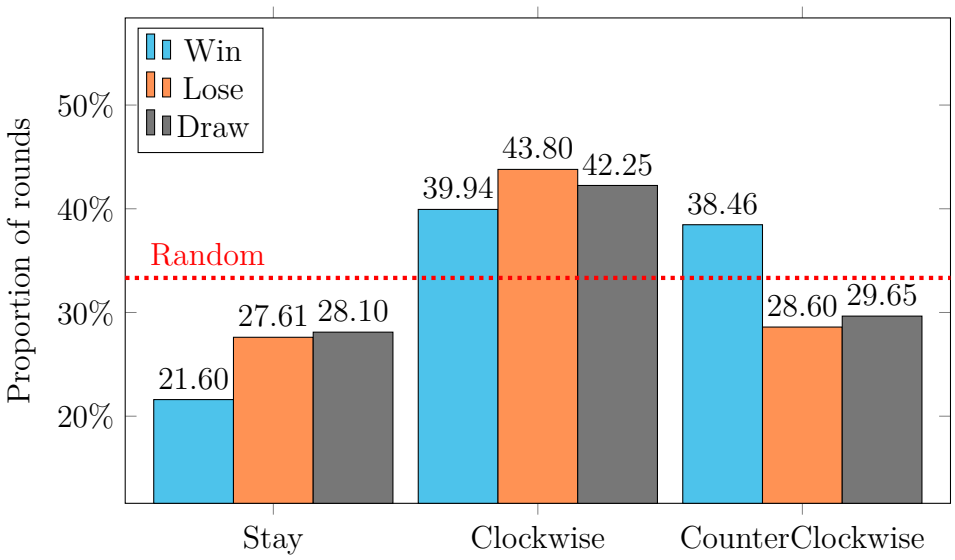


Figure 4.7.: Cycling behaviour of all users after winning, losing or drawing in the last round, expected uniform distribution marked in red

Both these factors actually imply that our users tried to counter the win-stay-lose-shift strategy. They intentionally do not stay after a win, while predicting their opponent to do so, therefore they cycle clockwise after their own loss. The probabilities for the remaining options (clockwise & counter-clockwise after a win, and Stay & counter-clockwise after a loss) do not differ decisively, which further supports the claim. The increased probability of the users to cycle clockwise after a draw shows, that the players expect their opponent to shift action when there was no winner in the last round.

A look at the bots cycling behaviour, which is visualised in Figure 4.8, shows many biases, which firstly implies a slightly imbalanced selection of bots. Secondly it can be used to explain the user behaviour.

Depending on the below or above average use of a specific cycle behaviour of the bots we can infer the best response (BR) of our players. The best responses for all cycle combinations are listed in the appendix in Table A.2. For example should the high probability of the bots to stay after a win correspondingly be met by the users with a comparable amount of clockwise rotations after a loss. These belong together, since intuitively bot win equals user loss. Interestingly the same behaviour is enforced after a bot loss and draw, as they tend to cycle clockwise or counter clockwise afterwards. For both the best response for the users is to cycle clockwise. This definitely allows us to explain the unusual high use of clockwise cycles in the players' behaviour.

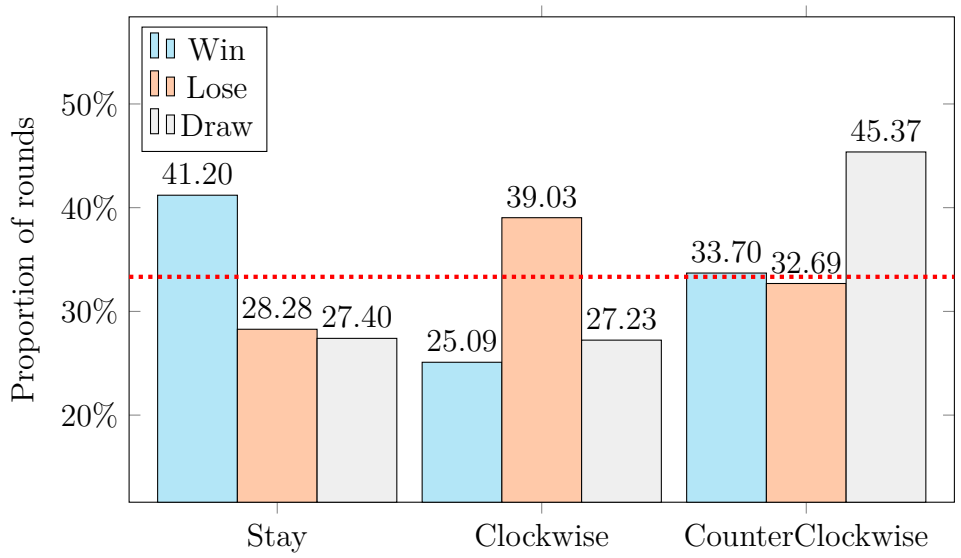


Figure 4.8.: Cycling behaviour of the bots after winning, losing or drawing in the last round, expected uniform distribution marked in red

We illustrate this by comparing the bots' cycle probabilities to the corresponding best responses for the players in Figure 4.9. It shows the respective deviation of the actions from the uniform distribution. It is important to note, that the deviation for the below average usage of the cycle 'BR-' was negated for visualization purposes. The exact numbers to this figure are listed in the appendix in Table A.3.

We see that the tendencies in the bots' cycle behaviour were answered very well by our participants. Solely the clockwise rotation after a draw was not met correctly by the players. The only other bad results (CC|W/L) are too close to the uniform distribution to be relevant. It is interesting to note, that many of the easier to interpret cycles, like staying after a win, were over-interpreted by the users. This reinforces the theory that the bot cycle tendencies have been detected by the participants.

Overall the comparison showed that the cycling behaviour of the human participants was clearly influenced by our chosen bots. Therefore we can not draw conclusions on human cycle preferences. We managed to show though that the cycle behaviour of many players' was directly influenced by their opponent's biased decision-making.

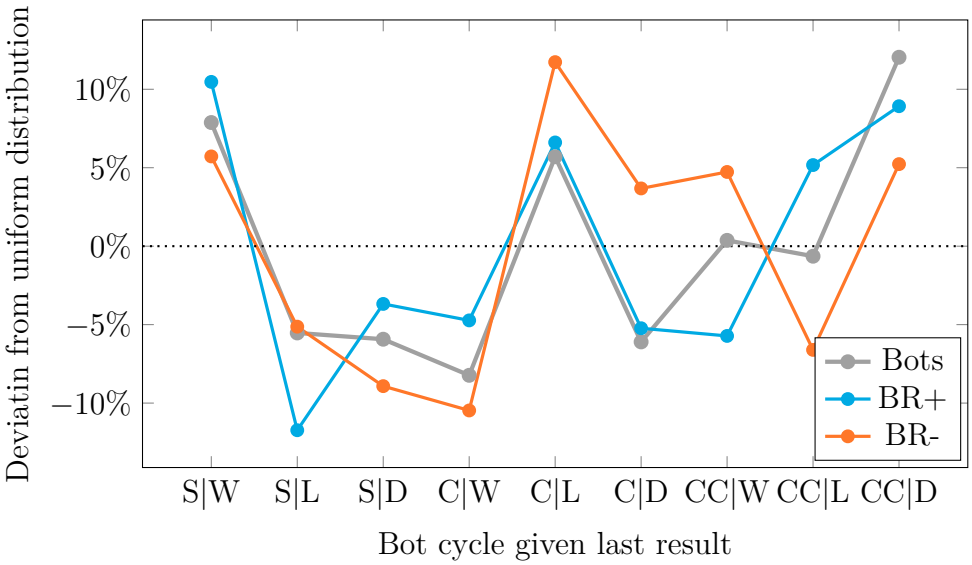


Figure 4.9.: Visualization of the user's response to the bot's cycle behaviour. Best response for above average usage of this cycle in green. Negated best response for below average usage in red

Continued cycling

When pursuing the cycle assumptions onto another turn, as Dyson et al. [16] propose in their work, we obtain the proportions visualized in Figure 4.10. We use the label 'maintain' for users who continue using the same strategy, meaning people who e.g. cycled clockwise after the initial round and continued doing so in the one after as well. 'Downgrade', when players adapt their cycling behaviour clockwise, e.g. staying in the first round and then cycle clockwise in the second, or cycling counter-clockwise and then stay. The last label 'Update' is defined accordingly, when players adjust their behaviour counter-clockwise. An exact list with all combinations and the corresponding label can be found in the appendix in table A.4. The figure shows a fairly balanced distribution of reactions, but with a very significant trend towards maintaining a strategy once started (38.40%, $p<.001$, $z=16.96$). This coincides well with the presented work of Dyson et al. and again shows an interesting peculiarity of human behaviour. This result also is not provoked by the bots secondary cycling behaviour, which shows a course very close to the uniform distribution (Figure A.1).

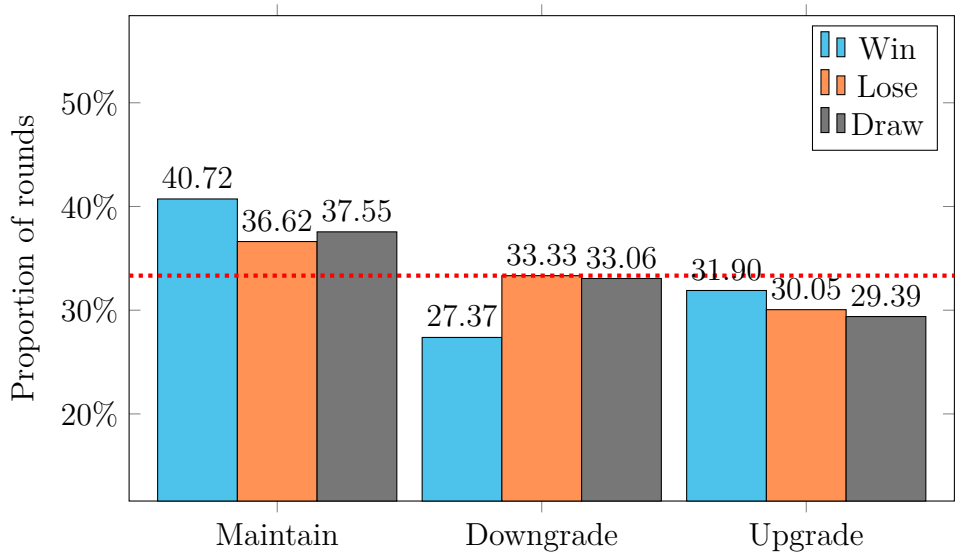


Figure 4.10.: Continued cycling two rounds after the initial win, lose or draw, expected uniform distribution marked in red

4.2.3. Questionnaires

We also conducted and evaluated almost 1600 questionnaires, held after every completed game. The most relevant questions we ask refer to the users own used strategy and the strategy he identified in his opponents play style. Figure 4.11 shows that over half of our users tried to read the opponents throws and react accordingly in their games. This supports the results from 4.2.2, where the cycle behaviour of the users was examined. Around 37.3% (NoStrat + Random + FixedPattern + CounterOwn) on the other hand claim that they ignored their opponents actions. The remaining opted for a strategy in between.
The strategies our participants detected in their opponent are more evenly distributed. When looking at the correctly given predictions listed in Table 4.2 we see though that only the Sequence Bots (Seq3, Seq4 & Seq5), corresponding to the 'Fixed Pattern' answer, were detected correctly above average. The average ratio in this case means that we select one out of the seven (excluding 'Other') possible answers in the questionnaire at random, leading to an average hit rate of 14.3%. The increased proportion for the Sequence Bots is not surprising, as a static repeating pattern is easier to detect than random movement or specific cycling behaviour. Moreover, the Sequence3/4/5 bots have a fairly high probability of 77.8%, 55.5% and 38.3%, respectively, to not even use all three available throws, which makes them even more predictable. As the 'GameFrequency Bot' has no direct answer allocated to its strategy, we assign 'Read Opponent' to it, leading to an above average hit ratio as well. This has to be seen in perspective though, as 'ReadOpponent' is the most used answer overall (27.8%) and also covers multiple other possible strategies, which means that the 'GameFrequency' strategy probably was not identified this often.

Overall we can say that the specific strategies were not reliably identified by our probands. Solely the fixed pattern strategy shows predictability.

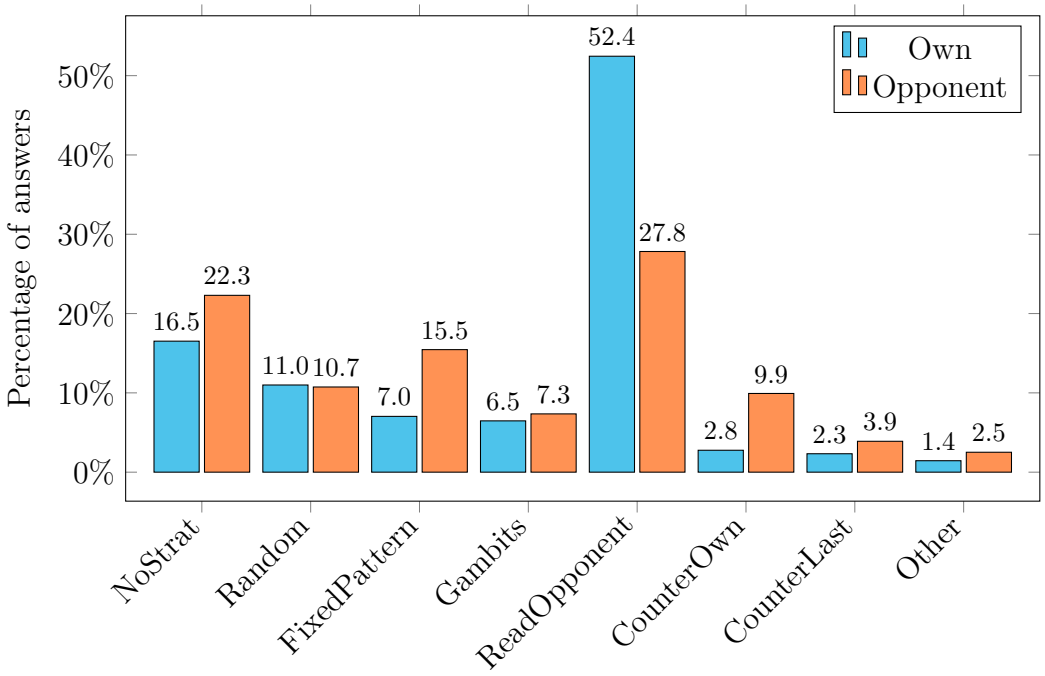


Figure 4.11.: Evaluation of answers given in the questionnaires referring to the own and the opponent's strategy choice

Table 4.2.: Number of played games and completed questionnaires for the different bots and the number and corresponding proportion of correctly identified strategy for the opposing bot

Bot	Random	Seq3	Seq4	Seq5	Counter Last	Counter Own	Game Freq	Overall
Games	378	67	379	289	342	424	394	2273
Questionnaires	256	33	281	210	256	285	271	1592
Corresponding Answer	Random	Fixed Pattern			Counter Last	Counter Own	Read Opponent	
Correct	19	10	55	45	6	29	76	240
Proportion	7.4%	30.3%	19.6%	21.4%	2.3%	10.2%	28.0%	15.1%
			21.0%					

4.3. Second Experiment

For the second experiment we collected the data of 53 more users, now including the cognitive reflection task as well as the different N-back tasks, described in 4.1.3. Descriptive statistics about the second experiment can be found in Table 4.3, the participant's demography is visualized in Figure 4.12. It is important to mention that we reduced and adapted the used bots for this experiment to make it more clear if a user identified a strategy. Now we only use the Sequence3Bot, as well as the CounterLastBot and CounterOwnBot. The latter with adjusted probabilities, now using their intended throw in 60%, from the 50% of the first experiment. The corresponding throw and cycling occurrences can be found in the appendix in Table A.5 and A.6, as they are not that important for this part.

Table 4.3.: Descriptive statistics about the second experiment

	Value	STD
Users	53	
Games	738	
Turns	9289	
Questionnaires	557	
Cognitive reflection tasks	53	
N-back tasks	53	
Average games per user	13.91	33.46
Average games per user (no top5)	5.06	9.28
Turns per game	12.59	2.24

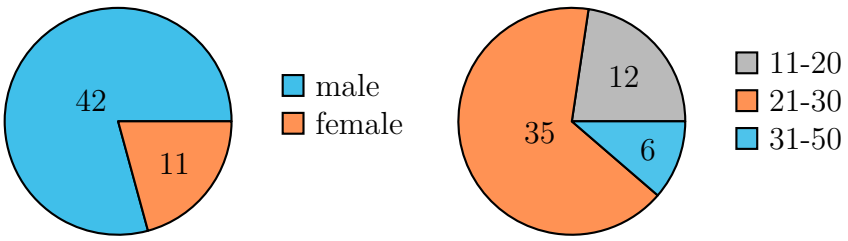


Figure 4.12.: Gender (left) and age (right) distribution of the users participating in the second experiment

4.3.1. Cognition tasks

Cognitive reflection task

As the cognitive reflection task, simply consists of questions with a right and multiple wrong answers, the evaluation is carried out by comparing the amount of errors a user makes. This is visualized in Figure 4.13. We can see that over half of the participants were able to perform some kind of reflection on their answers and return all questions correctly. It is worth to note that, even though participants who knew the task already achieved overall slightly better results, many (around 30%) still failed to pass the task error-free.

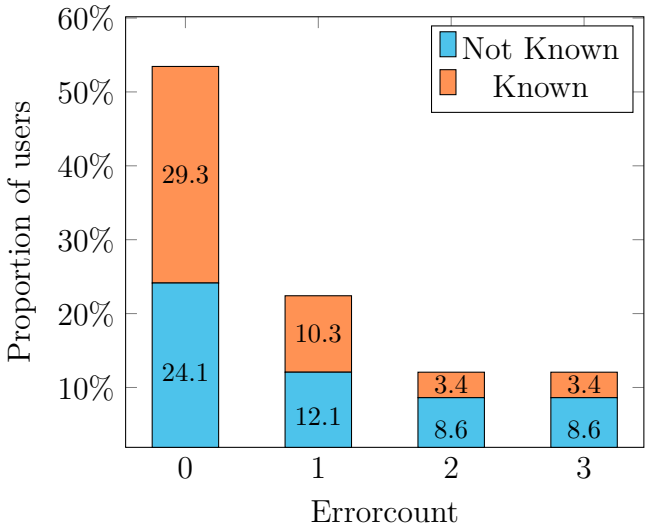


Figure 4.13.: Proportion of users making a specific amount of errors during the cognitive reflection task. Divided by users who already knew the task and those who did not

N-Back Task

The N-Back task is evaluated similarly, as any input is inevitably correct or wrong. Figure 4.14 shows the proportion of users making a specific amount of errors, grouped by the three different values for N. We see that for increasing N the proportion of users who achieved good results decreases rapidly. For N=1 almost 80% managed to stay below 2 errors, still a little over 60% for N=2 and not even 18% for the last task. The average error counts per user are visualized in Figure 4.15. For N=1 our probands make 0.82 errors on average. This number doubles already for the second task and more than doubles again for N=3, where we reach 4.38 average errors per user. The division into forgotten inputs (false negatives) and wrong inputs (false positives) allows us to determine where the errors come from. Especially the latter mark a state of confusion, showing that the user was not able to follow the sequence anymore. This value more than quadruples from N=2 to N=3. In combination with the enormous drop in users who could achieve a good ($\leqslant 1$ error) result, we can say that the average user was able to 'pass' the 2-back task, but not the subsequent.

4.3.2. Questionnaires

Evaluation of the questionnaires for the second experiment show a very similar image to the one of the first experiment. The exact numbers are listed in the appendix in Table A.7 and Figure A.2. The given answers again suggest that the specific behaviour of the opponent was not detected during the games, even with the simplified bot structure. The distribution also implies that the behaviour of the opposing bot does not decisively change the player's chosen strategy, nor the proportion of strategies the users identified in their opponent.

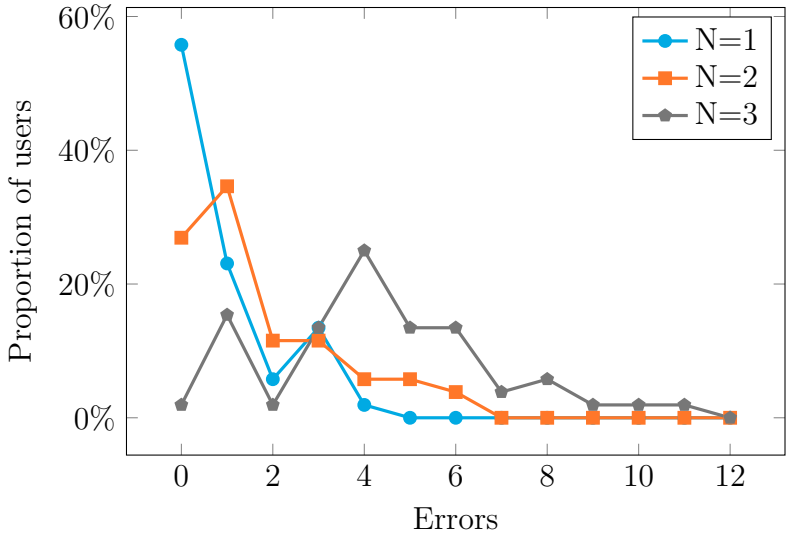


Figure 4.14.: Proportion of users making a specific amount of errors in the single N-back task for varying N

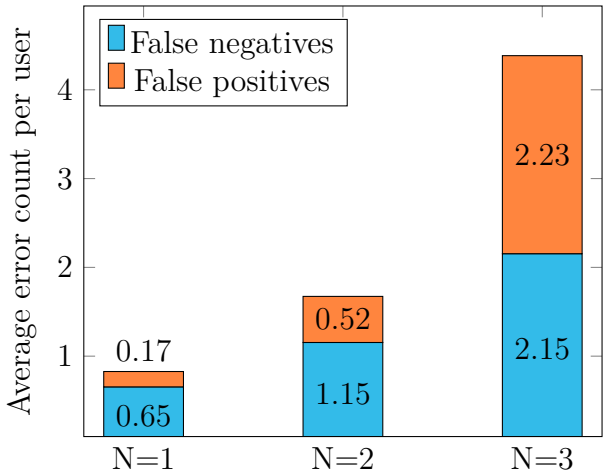


Figure 4.15.: Average Error count per user in the single N-back task for N=1,2,3, divided into forgotten inputs (false negatives) and wrong inputs (false positives)

5 Evaluation

Once the experiments were concluded, the unstructured data from the database is exported, structured and then used as the basis for interpretation and learning methods. This flow is visualized in figure 5.1 using a learning example.

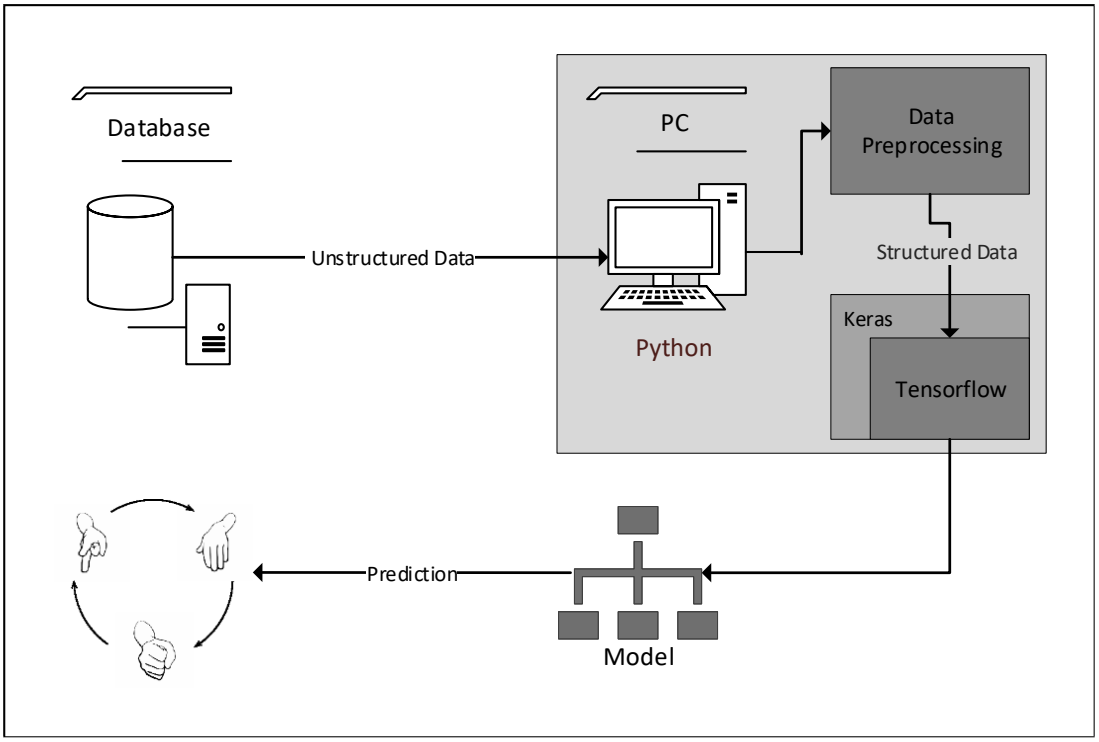


Figure 5.1.: Flow chart of events from the unstructured data to a prediction

5.1. RPS as Learning Problem

To be able to predict human behaviour in a game of RPS we have to formalize it as a machine learning problem. To begin with we use a single turn of our Rock-Paper-Scissors game as input, or more precisely, the selected action of the first player (A) plus the action of its opponent (B). The output at all times and for all methods is the predicted action, player A will use in the next turn. Formalizing the input/output pair like this:

$$A(t), B(t) \rightarrow A(t+1)$$

This also means, that we treat our problem as a *supervised learning problem*, as we use the real next action as label during the learning process.

5.2. Interpretation using Neural Networks

In all our networks we used a fully connected layer with three neurons and a *softmax* activation function as last layer before the output. Each of those neurons represent one of the three possible actions and the sigmoid function maps the conduct of all previous layers into an interpretable number between [0,1]. So to say, it shows us the proportion the network predicts this action to be the next one, eventually using the action with the highest value as output. The variable amount of layers in between have to be trained until they represent the data the best, as described in 3.3.1.

Used Software. All networks were trained using *Keras* [15], a high level deep learning library for Python. It builds upon Google's open-source software *Tensorflow* [1], one of the most common libraries for machine learning applications. For faster learning, we use Nvidia's *CUDA* [35] architecture to enable parallel CPU & GPU computations, running on a Nvidia GTX 1070 graphics card. Processing code was written using Python 3.6, running on Windows 10.

Training & Validation set. The training process was implemented using the *holdout method*, meaning that we randomly split our dataset in a training set, containing 80% of the data, and a validation set, containing the remaining 20%. During training the data from the first set is fed into the network, learning the corresponding parameters, e.g. the weights and biases. The validation set is held out during this phase, allowing us to later validate our model with new/unseen data. Like this we can assess the generalization performance of the model and also prevent overfitting.

Preprocessing. Additionally to bringing our data in the mentioned $A(t), B(t) \rightarrow A(t+1)$ format we had to enumerate the different actions 'Rock', 'Paper' and 'Scissors', so they can be interpreted by the network. A simple encoding into for example 1,2,3 brings some big problems, since if, e.g. the network fluctuates between action 1 and 3 it would probably predict a value around 2, a obviously wrong generalisation. Therefore we use so called *one-hot encoding*, which 'binarizes' the features. Transforming one feature with three values into three features with a binary value each (see Table 5.1), to prevent the unwanted dependencies. The same method is used for the user ids added during the input variation, later in this section.

Table 5.1.: One-hot encoding of the available actions to prevent internal dependencies

Action	Encoding
Rock	1 0 0
Paper	0 1 0
Scissors	0 0 1

5.2.1. Initialization with a genetic algorithm

To find the best model for our RPS problem we had to test multiple different combinations of parameters. To get an initial set of parameters we used a genetic algorithm [22], that tries to optimize the number of layers (or the network depth), the neurons per layer (or the network width), the activation function and the optimizer for the network. To do so it initializes the four parameters randomly from a given list to create a population of N networks. The configuration options are shown in Table 5.2. Each of the networks is then scored using a fitness function, in our case the prediction *accuracy* the network can achieve for our dataset. The top

performing networks and some random ones are kept, while the others get discarded. The random networks are kept to maintain diversity and to prevent getting stuck in a local maximum. The remaining networks get copied and mutated until we reach a population of N again. To do so, two random networks are chosen to 'breed' one or more new networks. The so emerged child will have some parameters of the first and some of the second parent network, mimicking natural evolution. This method leads to an optimized network significantly faster (around 85%) than the brute force method of trying all possible parameter combinations.

As our input structure is small, multiple network parameter combinations were able to achieve the same good results. Since the learning times increase drastically with higher numbers of neurons and especially layers, a goal is to minimize the model complexity while maximizing its performance. Therefore we choose the variant with 96 neurons in 1 layer as our starting point. As activation function, the *Rectified Linear Unit* (ReLU) performed same or better for all combinations, making it the obvious choice. The different optimizers for the error minimization using gradient descent (see Section 3.3.1), did not impact the result significantly, leading to the usage of Adam, as it is less complex. For the loss function, categorical cross-entropy suits our problem clearly better, as we formulated it as a classification problem.

Table 5.2.: Parameter combinations used as input for the generic algorithm, best combination marked in bold

Parameter	Selections
Neurons	2, 4, 8, 16, 32, 64, **96**, 128, 256
Layers	**1**, 2, 3, 4, 5
Activation function	Linear, **ReLU**, Tanh, Sigmoid
Optimizer	RMSprop, **Adam**, AdaMax, Nadam
Loss	**Categorical cross-entropy**, MSE

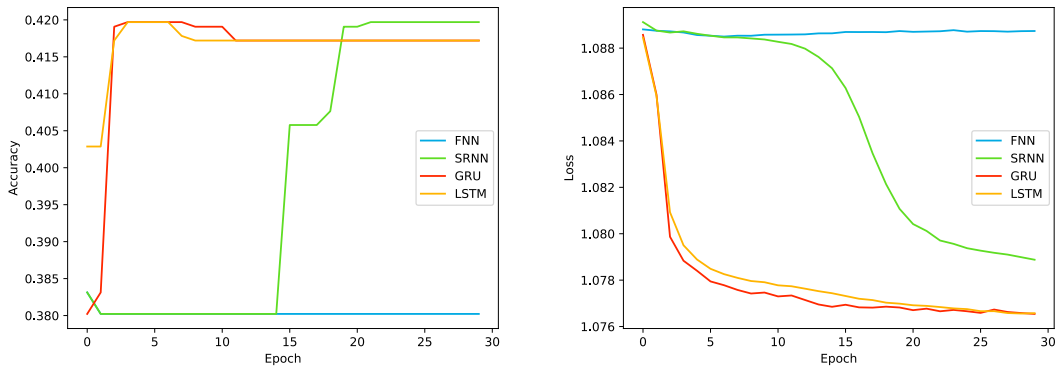
5.2.2. Network types

First of all we compared the different network types described in 3.3.1 and 3.3.2. It is important to emphasize that we did not optimize all four architectures separately, but optimized the fully connected Feed Forward Network and the LSTM with the genetic algorithm only. Since the variant with 96 neurons in one layer, ReLU activation and Adam optimizer achieved top performance for both, we adopted

those for the other two network types as well. This allows for a well balanced comparison, as the parameters coincide while only the network types vary. Table 5.3 depicts the corresponding accuracies, as well as the amount of epochs needed to reach this accuracy, for the four different network types. The results are averaged over five different random initializations of the training procedure, meaning the data is shuffled in different succession, resulting in different learning orders and validation sets, leading to more significant results and deviations. A descriptive course of the accuracy and loss for one fixed random seed, can be observed in Figure 5.2. It shows the learning process over the number of learning iterations, called epochs.

Table 5.3.: Comparison of the four used neural network types with fixed parameters, contrasting the maximum accuracy and the amount of epochs needed to reach it, averaged over multiple random seeds

Network type	Accuracy mean	Epoch
Feed Forward Network (FNN)	40.87 ± 0.32	7.6 ± 3.3
Recurrent Neural Network (sRNN)	41.87 ± 0.76	25.0 ± 4.0
Gated Recurrent Unit (GRU)	42.16 ± 0.66	3.6 ± 0.5
Long Short Term Memory (LSTM)	42.09 ± 0.75	4.8 ± 0.3



(a) Validation accuracy **(b)** Validation loss

Figure 5.2.: Accuracy and loss curve for different network types using the same parameters, reflecting one of the random seeds

The learning process for the recurrent network types can be easily traced. The validation loss drops continuously, until it converges towards a static value. This is mirrored onto the accuracy as well, where we get good values at the corresponding epochs. The small dip in accuracy afterwards for the LSTM and GRU network type can probably be attributed to overfitting, as the networks learn dependencies of the specific random seed, like the order of the input sequences. The feed forward network was not able to learn from the data. This can mean that the parameters were not set up correctly for this type, or that the peculiarities of the dataset in this succession can not be picked up by a FNN. When looking at the different random seeds the FNN always performed worst, but at least sometimes was able to learn some dependencies, leading to a slow drop for the loss function. Overall we can say that the contradictory nature of our problem did not fit the feed forward network structure.

In conclusion it is evident that the LSTM and the GRU outperform the simple recurrent as well as the standard feed forward network types. Notably all recurrent types exceeded the non-recurrent neural networks prediction accuracy, due to their inherent capability to model temporal data. This also demonstrates the additional dependencies that can be picked up during a game of Rock-Paper-Scissors.

The better learning times led to the usage of GRU as our continuation network type, as they converged faster than standard RNN and the LSTM, just as their internal structure suggests.

5.2.3. Sequence learning

A first interesting comparison arose when we increased the size of a single input interval. So instead of using the current input to predict the next action, we additionally used a variable amount of history steps n. Meaning for n=1 we still formalized our problem as

$$A(t), B(t) \rightarrow A(t+1).$$

For n=2, we added A(t-1) and B(t-1):

$$A(t-1), B(t-1), A(t), B(t) \rightarrow A(t+1)$$

This can be repeated for any n, as long as it does not surpass the length of a complete game. Figure 5.3 shows the accuracy means for a fixed GRU network with 96 neurons in 1 layer for n=1,...,8. To improve significance, again all results were averaged over five different random seeds of our dataset.

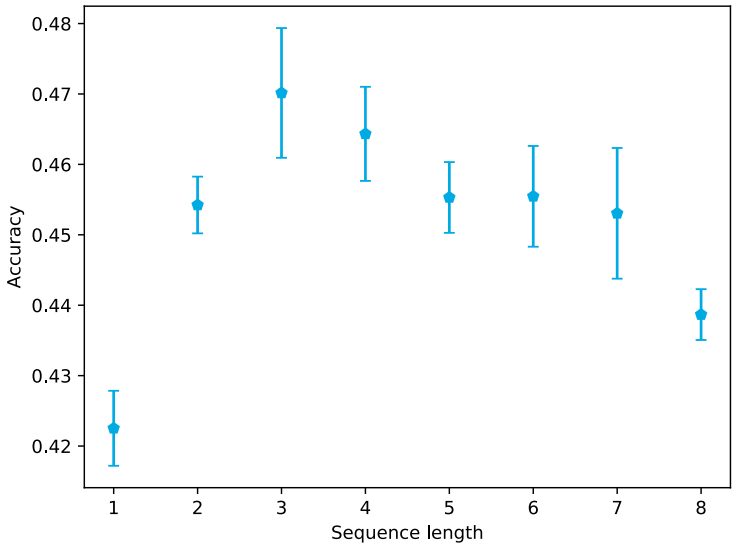


Figure 5.3.: Accuracy means and standard deviations for varying sequence length.

Interpretation. The figure shows clearly, that the accuracy improved for n>1. This is comprehensible, as the network is able to conduct more information out of the longer history sequences. It is easier to predict the next move, if you have additional background information, like if the player won or lost the rounds before and what action he used there. The best results were obtained for n=3,4. This is interesting, but still easily traceable as a big problem for $n \to k$ is, that the amount of datarows decrease linearly. For every game with length k, there exist exactly $k - n + 1$ datarows, that the network can use. Therefore, for bigger n, there is less data available. Additionally are longer sequences more rare, which makes it significantly harder to find matching sequences, which seemed to be a relevant aspect for our predictions. For n=1 the biggest amount of data is available and it is also easiest to find a gameround, where another user, or even better the same user in a different game, used exactly the same throw. As for n=1 this is only one of the known three actions, this is quite probable, but also means that there exist multiple matches, where the output differs. This let the network work with more information, but at the same time with contradictory information. For n=3,4 the two factors seemed to be balanced the best. Another idea that could have played into this result, is the fact that most humans are not capable to include

information that lies more than two steps back (see 4.3.1). A sequence length of 3 represents actually the current round plus the last two, reflecting exactly the just mentioned maximum steps. The spike after this maximum coincides well with the results from the N-back tasks.

5.2.4. Input variation

Lastly we tried to evaluate different networks by varying their input features. For this purpose we again fixed the network parameters, using a GRU with 96 neurons in 1 layer, as this structure showed to be the most effective, especially considering learning times. We then changed the input from the standard $A(t), B(t)$ to those listed in Table 5.4 and compare the maximum accuracy reached with this feature combination. This data is additionally visualized in figure 5.4.

Table 5.4.: Variation of features used as input for the neural networks

Input	Abbr.	Accuracy
A(t), B(t)	AB	42.17 ± 0.66
A(t)	A	42.15 ± 0.80
B(t)	B	35.97 ± 0.27
A(t), B(t), Result(t)	ABR	42.28 ± 0.89
A(t), Result(t)	AR	42.37 ± 0.78
B(t), Result(t)	BR	42.41 ± 0.75
A(t), B(t), Gender	ABG	42.90 ± 0.30
A(t), B(t), R(t), UserIdA	IdA	47.75 ± 0.30
A(t), B(t), R(t), UserIdB	IdB	42.28 ± 0.46
A(t), B(t), R(t), UserStatsA	StatsA	49.57 ± 0.46

Interpretation. It stands out, that all combinations that hold the same information, like AB, ABR, AR & BR, led to a basically equivalent accuracy slightly above 42%. When looking at the exact classifications we noticed that they coincide well with the allocations that occur when just always using the label that was predicted the most for this input combination. Meaning, that when labelling every 'Rock', 'Rock' input with 'Scissors' as the next action etc. we receive a similar prediction rate. This shows that for the small feature lists, the networks did not actually learn anything noteworthy. It seems they just picked up the global tendencies present in the data. This actually is evident, as with the randomization of the turns the sequence structure is lost and therefore also some of the peculiarities we examined before (first throw statistics and secondary cycling).

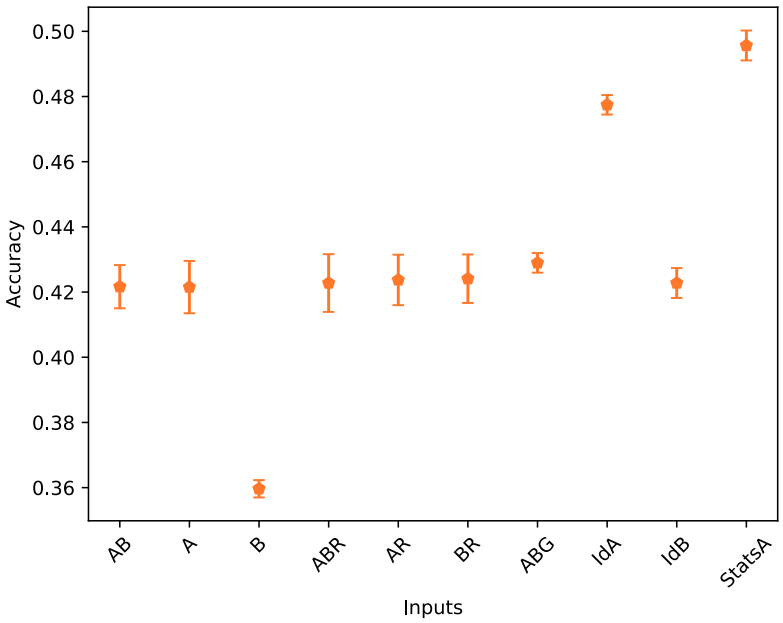


Figure 5.4.: Accuracy means and standard deviations for varying input features using a single layer GRU

Interestingly the input of player A's action alone led to a similar value as well. Notably this can not be explained with the global tendencies in the data anymore, as the tendency using A's action only, merely leads to an accuracy around 38.5%. This implies that the network was able to infer the same information from the chosen action of player A only, as from both players concurrently. This leads us to the assumption that for many players the opponent's action did not impact their future decision decisively. This initially seems to contradict the results we obtained in chapter 4.2.2, where we examined the users cycle behaviour. In fact though we showed that our participants reacted conditionally on their opponent's cycle behaviour. The used throw itself was no part of the dependency. This implies that the users picked up the cycle tendencies of their opponents conditioned on the result of the last round, the specific throw on the other hand did not impact their behaviour significantly.

A reassuring result unfolded when looking at input B, together with the IdB input. The average result for the latter and the only slightly above random for the first one shows that the opposing bots exact strategy mostly was not detected by

our participants, as the users behaviour did not change noticeable, when playing against the different bots. We can infer this, since changes in strategy dependant on the opponent is an exploitable change in behaviour, which the neural network would have picked up when fed with this additional information. This observation again also supports our assumption from the questionnaires, where only a really small portion of bot strategies have been identified.

Subsequently we considered different additional input features and their impact. Including the gender information increased the prediction accuracy only minimally by around 0.5%, but it decreased the deviation considerably. This means that gender did not really impact the behaviour, but the added information allows our network to classify some of the infrequently used throw combinations better. It is important to notice though that our experiments exhibited a rather biased demography, with below 25% female participants. The accuracy for those 25% seemed to only increase marginally though, leading to the presented result.

Adding the specific user id to every turn gave the network a significant edge. The accuracy rose by around 5.5%, as the network can now fall back on specific behaviour of every user. This implies that single users have reoccurring patterns that differ substantially from the average player. As final attempt we feed our network with additional statistics about the user. The exact composition includes the players throw distribution as well as his cycling behaviour after a win, lose and draw. This actually means that we add information to the network that was extracted from the complete dataset beforehand, meaning that we also include information from the validation set into training diminishing its expressiveness. It allows us to reach a prediction accuracy of almost 50% though, marking the highest value for this dataset for all used methods and inputs.

5.3. Interpretation using Recommender Systems

System adaption. Since our problem is no classical problem to solve with a recommender system, we had to adapt some of its characteristics. Firstly the items we want to rate are now the sequences of different length, consisting of the last n throws for both players. The output we want them to predict is, as for the other methods, the throw that comes next. Additionally we had to take into account, that the same user can rate the same sequence multiple times with different outputs, because they occurred multiple times in his games and he reacted differently. As there exist only exactly $3^2 \cdot n$ different sequences, and similar sequences represent totally different behaviour in the game, item-based filtering approaches do not offer valuable results.

This left us with user-based collaborative filtering concepts to represent the recommender methods. In the first approach, we ignored the fact for which user a given sequence occurred, but counted the times a specific output followed this sequence. The one with the most votes was then chosen as predicted next throw. In case of two- or three-way ties, we added this sequence two to three times with the different outputs. Afterwards we verified for each sequence if the predicted result coincided with the actually observed ones, leaving us with a prediction accuracy similar to the ones issued by the neural networks. Per above description this technically is still a user-based approach, as we compare how other players react in the same situation. Since it is heavily centred around the given item/sequence though, we will refer to it as sequence-based in the continuation of this thesis. Pseudo-code to this approach is outlined in Algorithm 1.

In our second approach we addressed the single user more, giving every user exactly one vote on what the current candidate should do. If for any user the given sequence occurred multiple times with different outputs, we had an intern vote again, similar to the one described in the first approach, to determine his final answer. The answers again get verified, leaving us with a prediction accuracy.

It is important to note that for both recommender approaches we did not hold out any data to validate the model, as this is not needed for this type.

Algorithm 1 Sequence-based collaborative filtering

```
1: for sequenceA in data do
2:     voteRock=votePaper=voteScissors=0
3:     for sequenceB in data do
4:         if sequenceA==sequenceB then
5:             if predictionB==Rock then
6:                 voteRock+=1
7:             else if predictionB==Paper then
8:                 votePaper+=1
9:             else if predictionB==Scissors then
10:                voteScissors+=1
11:    if (voteRock+votePaper+voteScissors) > 0 then
12:        for action in (Rock,Paper,Scissors) do
13:            if action has most votes then
14:                if predictionB==action then
15:                    correct+=1
16:                else
17:                    incorrect+=1
   return correct/(correct+incorrect)
```

Interpretation. Using the two recommender system approaches, we reached the accuracies displayed in figure 5.5. It stands out that the best performance is achieved using only one set of inputs, and the around 45% for both approaches exceed the expectations. It seems that many users react very similar in comparable situations, as well as similar to other users in the same situation. This was also represented in the primary cycling behaviour, examined in section 4.2.2.

The sequence-based approach expectedly outperformed the user-based approach in all scenarios. This is not surprising as the amount of votes is significantly higher (thousands compared to exactly 185-1), reducing outliers and improving the inferred prediction accuracy all together. Since the users only voted on other peoples behaviour, this also means that reoccurring sequences in a single user are not taken into account. The significant drop in accuracy from sequence length one to two implies that secondary behaviour, meaning reactions that do not occur as direct response are definitely less clear. This coincides with the secondary cycling behaviour we examined in 4.2.2, where the cycling probabilities did not show peculiar outliers. The even higher drop in the user-based case demonstrates that the same individual seems to repeat behaviour while different users show only

little affinity. This is somewhat balanced out again for n≥3, which implies that the secondary cycling behaviour is particularly diverse for the different users. The overall descending accuracy for growing input length can again be attributed to the linearly decreasing amount of data, paired with an increase of input combinations by a factor of $2n$. This consequentially leads to a lack of sequence matches, which is the only information our recommender system can use.

It is also worth mentioning that the sequence-based approach with sequence length four actually reflects the exact strategy used by the NYTimes, described in the Related work chapter.

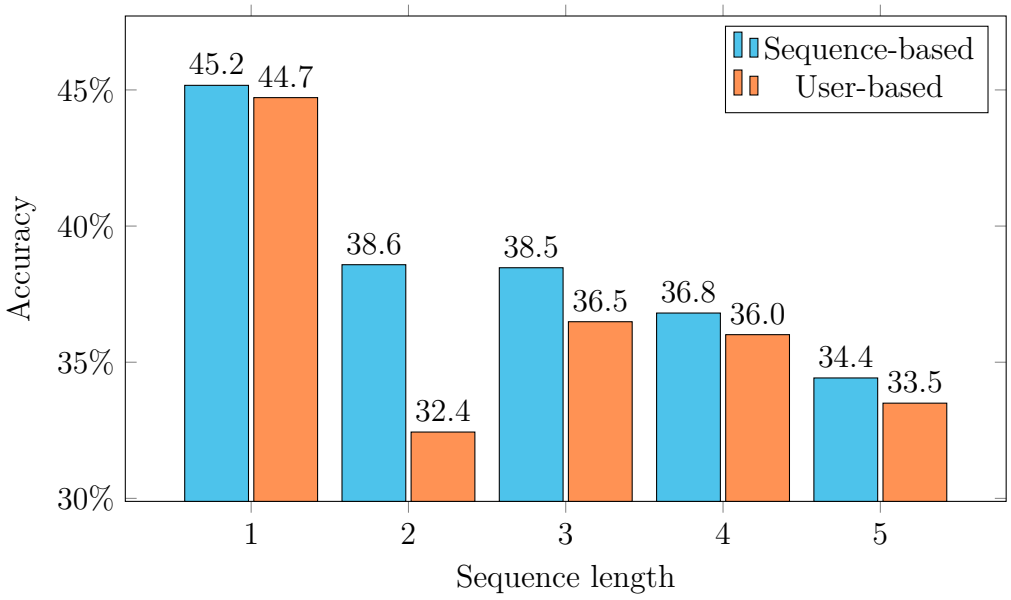


Figure 5.5.: Accuracy reached with the two different recommender system approaches for varying input length

5.4. Interpretation using Support Vector Machines

We trained our SVM using *LIBSVM* [13], one of the most prominent libraries for support vector machines. By using grid search and cross validation techniques we were able to compare different parameter combinations. The best are then used to classify our data by dividing the space with multidimensional hyperplanes. The chosen kernel, the parameters and the five-fold cross validated prediction accuracy, are listed in table 5.5. As input for the machine we use the classic A(t), B(t), with A(t+1) now being the class label we want new data to be allocated to.

Table 5.5.: Parameter combination and values leading to the best accuracy for our support vector machine approach

Kernel	Parameters	Accuracy
RBF	$C = 0.125, \gamma = 2$	42.272%

Interpretation. A vivid representation of the data and corresponding classification through the SVM is presented in figure 5.6. The light gray dots depict the different input combinations we can have. As we only have two inputs with three possible values each, this adds up to nine different data piles only. But each of them is thousands of data points deep, as 28000 rows of game data distribute over them. This makes our problem a very unique one to solve with a support vector machine. Since the thousand data points at every position also vary in the label/class they belong to, which in our case again is the class/throw that will be used in the following turn. The SVM now has to find a partitioning of the complete plane into the different classes depending on the amount of data and the distribution of classes on the different positions. The resulting image is easy to interpret, as the SVM simply predicts the label of a new data point dependant of its position in the plane. A entry with action A = 'Scissors' and action B = 'Scissors' for example will be labelled as class 'Paper', as this point origins in the orange area. With this technique the method achieves a prediction rate of 42.27%, which is a reputable result, that can compete with neural networks for this simple input structure. The strict labelling of the input combinations actually represents the global tendencies we mentioned in the interpretation of the input variation section earlier this chapter.

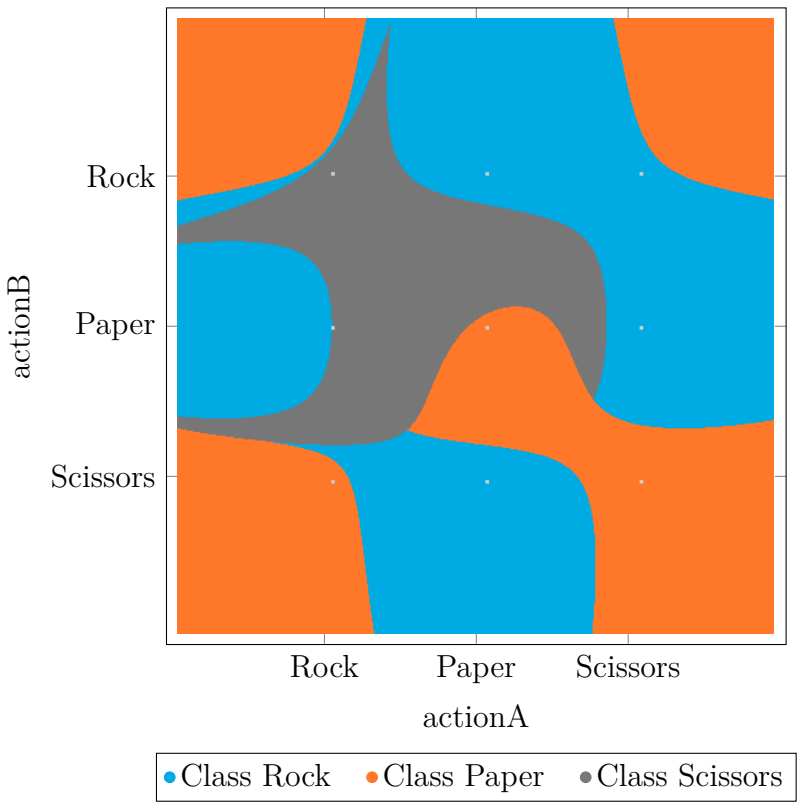


Figure 5.6.: Visualized classification as performed by the SVM, with the classes representing the predicted next throw. Single data points (overlapping thousands of times) shown in light gray

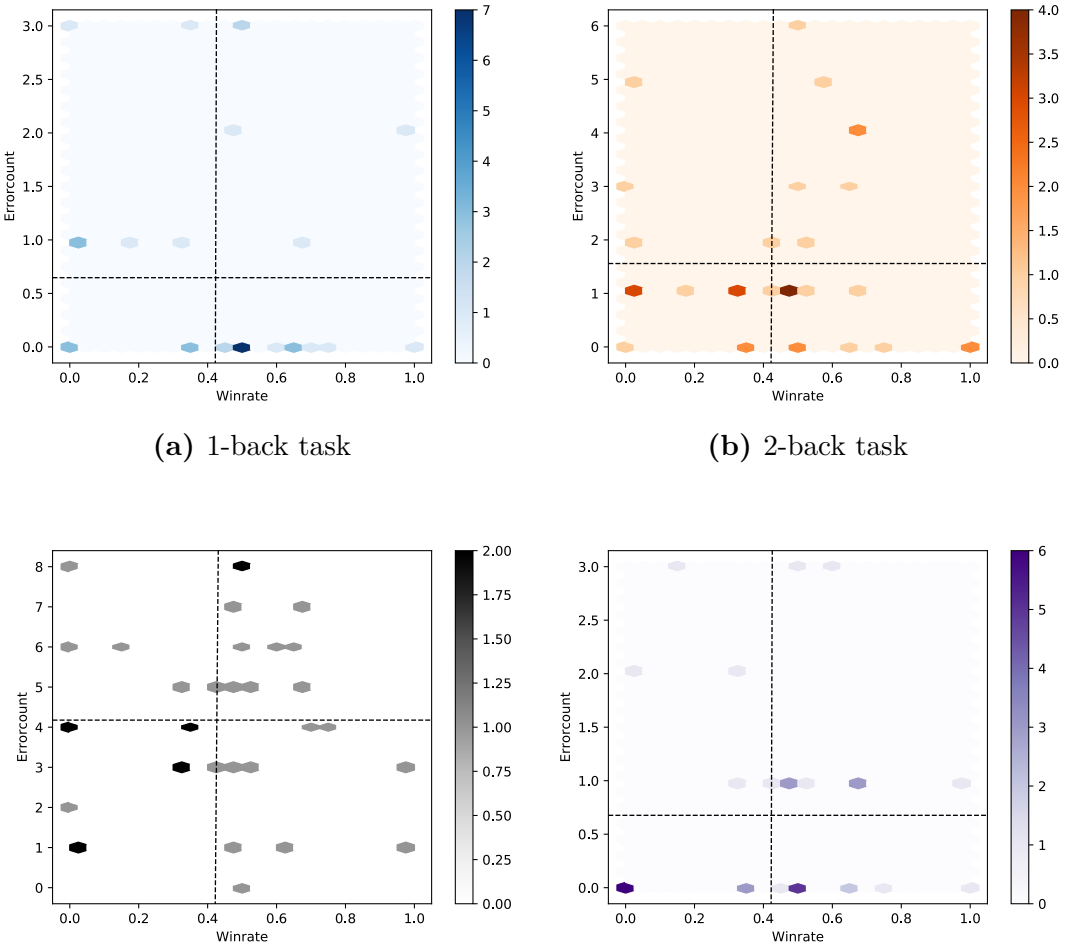
5.5. Cognition Tasks

Additionally we evaluated how different performance in the cognition tasks linked to the users behaviour in a RPS game. Figures 5.7a - 5.7d visualize the relations between the amount of errors a specific user made during the different cognition tasks and his winrate during all of his RPS games. The dotted lines represent the average performance in both. This division allows us to separate the data plane into four quadrants, each representing an above/below average performance in the respective fields. It is important to note that we could only use the data of 34 users for this calculations, as not all tests were complete or the associated users did not play enough games (≥ 2) afterwards. The average winrate for those equals 42.25%. This 'low' number emerges from the fact that the win rates are weighted equally, independent of the amount of games they are build upon.

Interpretation. As we can see, the distribution of data points is relatively balanced for all of the tasks, implying that there is no strong correlation between the two variables. We also notice though, that most points are in the bottom right quadrant, meaning that users with high win rates also tend to make fewer errors. This in combination with the amount of points in the top left corner, most apparent to see in 5.7a, suggests a small negative correlation between the variables. The exact numbers of this relation can be looked up in table 5.6. The values show that the correlation between the amount of errors in the n-back tasks and the players' performance in the games is consistently negative, but the numbers are too close to zero, meaning that this is not a significant discovery. Similarly the correlation with the CRT, where the values are small, but positive. This shows that we can not infer any information from the used cognition tasks to the performance in our RPS game.

Table 5.6.: Covariance and correlation between the errors in the cognition tasks and the winrate of the corresponding user

	1-back	2-back	3-back	N-back	CRT
Average error count	0.6471	1.5588	4.1765	6.3824	0.6765
Covariance	-0.0340	-0.0305	-0.0038	-0.0683	0.0167
Correlation	-0.1214	-0.0693	-0.0064	-0.0729	0.0659



(a) 1-back task

(b) 2-back task

(c) 3-back task

(d) Cognitive reflection task

Figure 5.7.: Illustration of the relation between the errors in the cognition tasks and the winrate of the corresponding user, darker colors represent overlapping data, dotted lines mark the average

6 Conclusion

In this thesis we tried to show that human behaviour in games can be predicted using computational tools. With a self acquired big data collection and machine learning methods we tried to find and predict peculiarities in the players' decision-making. By comparing different interpretation methods and the information they use, we hoped to evaluate the different features and get insight into human cognition.

We started this work with a theoretic analysis of human behaviour in decision-making scenarios and especially in Rock-Paper-Scissors. We showed that rational agents with perfect knowledge should chose their actions at total random, as this is the only strategy that prevents you from being predictable and therefore exploitable. It has been proposed though that humans are neither completely rational nor are they able to utilize perfect information nor are they able to perform actual random actions. It should correspondingly be possible to predict human behaviour in games using computational methods.

In order to examine this we conducted two experimental studies. On the basis of 28000 rounds of RPS, versus probabilistic bots, 2100 questionnaires and around 200 simple cognition tasks, we tried to gain further insight into peculiarities of human decision making. Evaluating the data, we were able to detect significant deviations from rational behaviour within our test persons. We identified characteristics such as the biased choice of throw, overall as well as especially in the first round of a game. Also the primary and secondary cycling behaviour of humans after winning, losing or drawing in a repeated game revealed interesting peculiarities. Those findings coincide well with earlier work on this topic.

During evaluation of the primary cycle behaviour we noticed a bias in the cycling behaviour of the bots. We managed to show that many of our users were able

to detect this deviation from rationality and counter it accordingly. As a matter of fact those deviations were even overcompensated by the users, adding another usable feature to the list of peculiarities.

When interpreting the questionnaires of both experiments, we found out that more than half of the users tried to read and react to the opponent's actions. This relates to the just mentioned dependencies in the cycle selection. On the other hand they also show that a third of the players intentionally tried to ignore their opponents strategies. The comparison of the questionnaires building on the two different experiments implied that the specific strategies of the underlying bots were not detected and that the player's choice of strategy is not impacted considerably by the opponents strategy.

All these discoveries were then used to create various models, which represent and predict human behaviour in a Rock-Paper-Scissors game. This was done using different machine learning methods such as neural networks, support vector machines and collaborative filtering approaches, a subgroup of recommender systems. All of them were compared and interpreted by evaluating their ability to correctly predict the next action of our players. For the basic evaluation, we used the chosen action of the opposing players in the current round and tried to predict the next action of the first player. This corresponds to a sequence length of one, a condition that was adapted in later stages of the evaluation by adding a variable amount of previous turns to this representation.

Formalizing RPS as a supervised learning problem held several difficulties. Mainly is the input feature count relatively small due to the simple structure of the game. This also leads to overlapping of the input-output combinations, which means that the same input can lead to contradictory outputs. These factors definitely limit the prediction accuracies the different methods can achieve.
The first representation we want to mention here was obtained using a support vector machine. This method tried to divide the data into different planes. Correspondingly are equal input combinations always classified to the same output. The method so to say sets up static rules about human behaviour in a RPS game. Prediction using this classification technique led to an accuracy slightly above 42%. This represents the global tendencies that are present in our dataset.

The straightforward recommender approaches, which mostly just count how other users react in similar situations, achieved a surprisingly good prediction accuracy of 45%, using a single sequence input. It shows how the overall throw distribution together with primary cycle behaviour alone, since this is the only information the recommender system is able to pick up, already represents the behaviour of

other users considerably well. It also proves that these behavioural characteristics reoccurred in multiple test persons. For increasing sequence length the prediction accuracy decreased continuously.

The results for the neural networks were as multifaceted, as the possibilities of this technique. After we found a suitable network structure for our problem and satisfactory parameters, we started by evaluating the standard inputs. Due to the randomization during learning, we lost many of the discovered peculiarities in the data, leading to a prediction score, similar to the SVM. By varying the input features of our networks, we were able to partially prevent this improve their results considerably. Firstly we changed the length of the input sequence we fed into the network. Adding these sequential dependencies, allowed the network to learn on more and less contradictory information. The network performance peaked for $n = 3$ at around 47%. This implied, that history more than 3 turns away did not add more behavioural dependencies. Which means that human behaviour in a game of RPS, that lies back more than three turns, does not influence the current decision considerably anymore. This coincided well with the results from the n-back tasks we conducted, which clearly showed that most humans are not capable to remember occurrences more than three steps away.

Further insights arose when changing the used input features for the neural network as a whole. First of all, we were able to support the claim that the average player did not identify the opposing bots strategy in their games, by adding said information to the network inputs. As this did not increase the information gathering performance of the network, we could infer that the users' behaviour did not change considerably facing the different bots. This also implied that the opponent's choice of action did not have as big of an impact on a player's decision making as expected. Indications for this were also found in the questionnaires and in the unaffected prediction accuracy when reducing the input to the action of player A alone. This should not be mistaken as statement that players play independent of their opponent. It just shows that the action itself is less important than the previous result, as the players behaviour is best portrayed using the cycle behaviour.

The overall best accuracies were achieved by adding the unique id of the user (47.7%), or when adding additional stats about the players, like their throw preferences or cycle behaviour (49.5%). This suggests that single users show reoccurring preferences, that can be predicted fairly well, but even out in the mass of the general population.

Comparing the different methods, we noticed that neural networks did not outperform the other two for small input feature sizes. This is evident as the strength

of artificial neural networks lies in solving complex problems with many features. A glimpse of this was recognized when we increased the available information and reached prediction rates for this dataset close to 50%. This is a reputable performance for this contradictory problem. Most importantly though this score should lie just above 33%, if the experiments were conducted by rational agents on both sides. We were therefore evidently able to predict human behaviour in games using computational tools.

In this thesis we presented a new big data set describing peculiarities in human behaviour while playing the game of Rock-Paper-Scissors. We demonstrated the behavioural peculiarities in human decision-making, and showcased the deviation from rational behaviour. We additionally detected, that most players in this game try to analyse their opponent's strategy and were able to predict deviations from rationality in the long term. The same was attempted and successfully accomplished using different computational methods. With the means of machine learning we were able to predict human behaviour in a game of RPS with far above random accuracy.

6.1. Future Work

In order to build upon these results, we can work on various properties. During examination of the experiments we noticed some flaws in their structure, which decreased the significance to cognitive science. First of all should the experiment be conducted with human players on both sides. We hoped that the bot structure allowed for easier modelling and to generate interesting changes in the decision-making process of the human players. Unfortunately, this was not the case and the bots' strategies mainly led to a distortion of the human behavioural peculiarities. The second point is to aim for a more balanced demography.

When continuing working on the current dataset, we could add further examinations. An interesting feature we could investigate is whether and how the cycle behaviour of users who claimed to have played strategies independently of their opponents differed from the others. Continuing on the interpretation methods we could firstly extend the recommender systems, by adding a similarity function to the users. This would enable an additional approach where only specific users are allowed to vote, probably leading to improved results and interesting interpretations.

Further ideas for the neural networks would be to combine the most successful

approaches regarding sequence length with the additional inputs. Secondly would it be beneficial if we found a way to insert information about the sequence of the turns, while keeping the context of a complete game. This could, for instance, be tackled using batch learning.

The last improvement could be to build a bot, which falls back onto the learned dependencies. By doing so the relevance of all inferred characteristics of the users could be tested and improved simultaneously. This would also allow us to actually compare the performance of this prediction approach with other proposed models.

A

Appendix

A.1. First Experiment

Table A.1.: First throw distribution of the bots in the first experiment.

	Rock	Paper	Scissors
Proportion	32.19	35.15	32.66

Table A.2.: List of bot throws and corresponding best response of the users for above and below average usage of the given throw

Bot throw given last result	Best response above average	Best response below average
S | W	C	S
S | L	S	CC
S | D	CC	C
C | W	CC	C
C | L	C	S
C | D	S	CC
CC | W	S	CC
CC | L	CC	C
CC | D	C	S

Table A.3.: List of bot cycles with the corresponding best response of the users for above and below average usage of the given action. Comparing deviations from the uniform distribution U to detect dependencies between bot and users cycle behaviour

P(Bot cycle given last result)	$P-U$	BR+ if $P-U>0$	$prob(BR+)-U$	BR- if $P-U<0$	$prob(BR-)-U$
S | W = 41.21%	+7.88%	C	**+10.47%**	S	-5.72%
S | L = 28.28%	-5.53%	S	-11.73%	CC	**+5.13%**
S | D = 27.40%	-5.93%	CC	-3.68%	C	**+8.92%**
C | W = 25.10%	-8.23%	CC	-4.73%	C	**+10.47%**
C | L = 39.03%	+5.70%	C	**+6.61%**	S	-11.73%
C | D = 27.23%	-6.10%	S	-5.23%	CC	**-3.68%**
CC | W = 33.70%	+0.37%	S	**-5.72%**	CC	-4.73%
CC | L = 32.69%	-0.64%	CC	+5.17%	C	**+6.61%**
CC | D = 45.37%	+12.04%	C	**+8.92%**	S	-5.23%

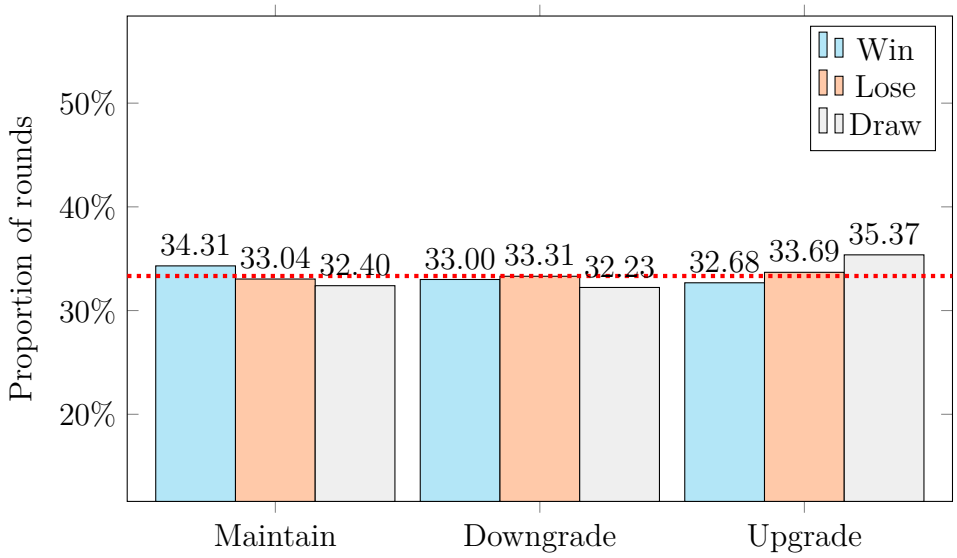


Figure A.1.: Continued cycling two rounds after the initial win, lose or draw, as performed by the bots

Table A.4.: Combinations and corresponding label for the continued cycling behaviour of our probands in the first experiment.

1st cycle	2nd cycle	Label
Stay	Stay	Maintain
Clockwise	Clockwise	Maintain
CC	CC	Maintain
Stay	C	Downgrade
C	CC	Downgrade
CC	S	Downgrade
Stay	CC	Upgrade
C	S	Upgrade
CC	C	Upgrade

A.2. Second Experiment

Table A.5.: Throw-distribution over all turns and users in the second experiment

Throw	Rock	Paper	Scissors
Proportion	0.3346	0.3400	0.3253

Table A.6.: Cycle distribution over all turns and users in the second experiment

	Stay	Clockwise	Counter-clockwise
Win	0.1220	0.4267	0.4513
Draw	0.3160	0.4214	0.2626
Loose	0.2139	0.5789	0.2072

Table A.7.: Number of played games and completed questionnaires for the different bots and the number and corresponding proportion of correctly identified strategy for the opposing bot in the second experiment

Bot	Sequence3	CounterLast	CounterOwn	Overall
Games	253	226	259	738
Questionnaires	193	169	195	557
Corresponding Answer	Fixed Pattern	CounterLast	CounterOwn	
Correct	39	9	1	59
Proportion	15.42%	5.3%	4.25%	10.59%

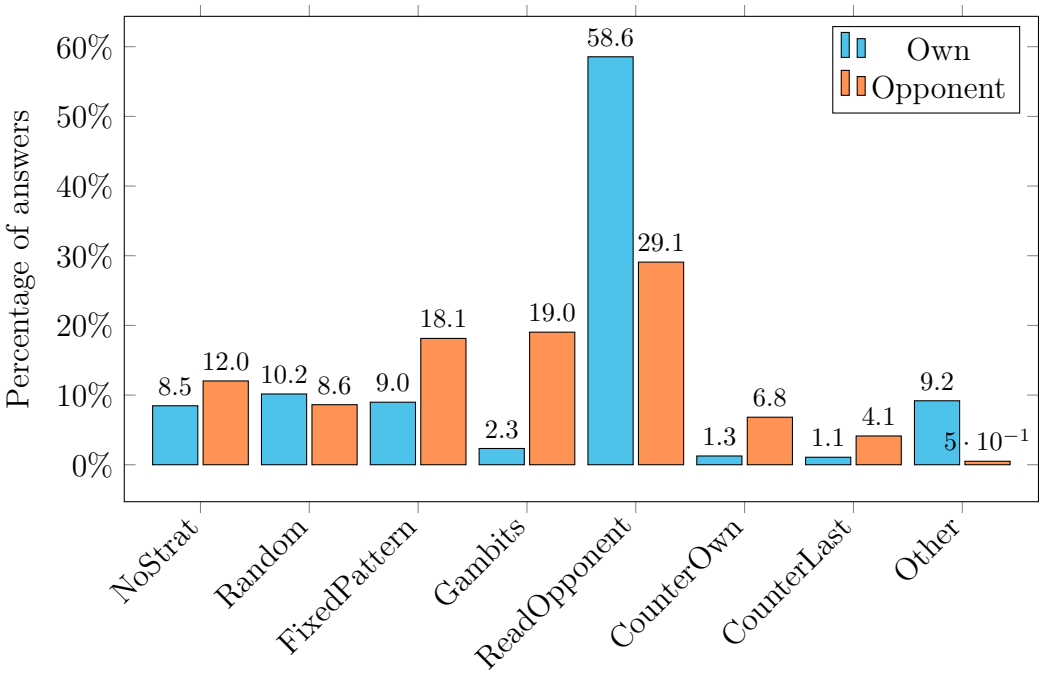


Figure A.2.: Evaluation of answers given in the questionnaires referring to own and opponents strategy choice for the second experiment

Bibliography

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

[2] Mark A Aizerman. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and remote control*, 25:821–837, 1964.

[3] Fathelalem F Ali, Zensho Nakao, and Yen-Wei Chen. Playing the rock-paper-scissors game with a genetic algorithm. In *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, volume 1, pages 741–745. IEEE, 2000.

[4] Maurice Allais. L'extension des théories de l'équilibre économique général et du rendement social au cas du risque. *Econometrica, Journal of the Econometric Society*, pages 269–290, 1953.

[5] Alexandr Andoni, Rina Panigrahy, Gregory Valiant, and Li Zhang. Learning polynomials with neural networks. In *International Conference on Machine Learning*, pages 1908–1916, 2014.

[6] Kwangyeol Baek, Yang-Tae Kim, Minsung Kim, Yohan Choi, Minhong Lee, Khangjune Lee, Sangjoon Hahn, and Jaeseung Jeong. Response randomization of one-and two-person rock–paper–scissors games in individuals with schizophrenia. *Psychiatry research*, 207(3):158–163, 2013.

[7] Shawn Bayern. Rock, paper, scissors: Humans against ai. http://www.essentially.net/rsp/index.jsp [Online], 03 2001.

[8] Sudeep Bhatia and Russell Golman. A recurrent neural network for game theoretic decision making. In *Proceedings of the Cognitive Science Society*, volume 36, 2014.

[9] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

[10] John S Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 43–52. Morgan Kaufmann Publishers Inc., 1998.

[11] Colin F Camerer. *Behavioral game theory: Experiments in strategic interaction*. Princeton University Press, 2011.

[12] Colin F Camerer, Robin M Hogarth, David V Budescu, and Catherine Eckel. The effects of financial incentives in experiments: A review and capital-labor-production framework. In *Elicitation of Preferences*, pages 7–48. Springer, 1999.

[13] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[14] Gary Charness and Matthew Rabin. Understanding social preferences with simple tests. *The Quarterly Journal of Economics*, 117(3):817–869, 2002.

[15] François Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.

[16] Benjamin James Dyson, Jonathan Michael Paul Wilbiks, Raj Sandhu, Georgios Papanicolaou, and Jaimie Lintag. Negative outcomes evoke cyclic irrational decisions in rock, paper, scissors. *Scientific reports*, 6, 2016.

[17] Seymour Epstein, Abigail Lipson, Carolyn Holstein, and Eileen Huh. Irrational reactions to negative outcomes: Evidence for two conceptual systems. *Journal of personality and social psychology*, 62(2):328, 1992.

[18] Shane Frederick. Cognitive reflection and decision making. *The Journal of Economic Perspectives*, 19(4):25–42, 2005.

[19] Herbert Gintis. Behavioral game theory and contemporary economic theory. *Analyse & Kritik*, 27(1):48–72, 2005.

[20] David Goldberg, David Nichols, Brian M Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

[21] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[22] Matt Harvey. Let's evolve a neural network with a genetic algorithm. *coastline automation, posted on April*, 2017.

[23] Sepp Hochreiter and Jürgen Schmidhuber. Lstm can solve hard long time lag problems. In *Advances in neural information processing systems*, pages 473–479, 1997.

[24] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. A practical guide to support vector classification. 2003.

[25] Samuel H Huang and Mica R Endsley. Providing understanding of the behavior of feedforward neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 27(3):465–474, 1997.

[26] Benjamin Kerr, Margaret A Riley, Marcus W Feldman, and Brendan JM Bohannan. Local dispersal promotes biodiversity in a real-life game of rock–paper–scissors. *Nature*, 418(6894):171–174, 2002.

[27] Wayne K Kirchner. Age differences in short-term retention of rapidly changing information. *Journal of experimental psychology*, 55(4):352, 1958.

[28] Benjamin C Kirkup and Margaret A Riley. Antibiotic-mediated antagonism leads to a bacterial game of rock–paper–scissors in vivo. *Nature*, 428(6981): 412–414, 2004.

[29] Daeyeol Lee, Benjamin P McGreevy, and Dominic J Barraclough. Learning and decision making in monkeys during a rock–paper–scissors game. *Cognitive Brain Research*, 25(2):416–430, 2005.

[30] Lola L Lopes and Gregg C Oden. Distinguishing between random and non-random events. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 13(3):392, 1987.

[31] John McCarthy, Marvin L Minsky, Nathaniel Rochester, and Claude E Shannon. A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955. *AI magazine*, 27(4):12, 2006.

[32] KM Miller, CC Price, MS Okun, H Montijo, and D Bowers. Is the n-back task a valid neuropsychological measure for assessing working memory? *Archives of Clinical Neuropsychology*, 24(7):711–717, 2009.

[33] Michael E Moore and Jennifer Sward. *Introduction to The Game Industry (Game Design and Development Series)*. Prentice-Hall, Inc., 2006.

[34] Allen Newell, Herbert Alexander Simon, et al. *Human problem solving*, volume 104. Prentice-Hall Englewood Cliffs, NJ, 1972.

[35] John Nickolls, Ian Buck, Michael Garland, and Kevin Skadron. Scalable parallel programming with cuda. *Queue*, 6(2):40–53, 2008.

[36] Michael A Nielsen. Neural networks and deep learning, 2015.

[37] Christopher Olah. Understanding lstm networks. *GITHUB blog, posted on August*, 27, 2015. URL `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`.

[38] CHETPRAYOON Panumate, Hiroyuki Iida, and Jean-Christophe Terrillon. A game informatical analysis of roshambo. 2016.

[39] Neil Pomerleau. Rock paper scissors.

[40] Gabriele Pozzato, Stefano Michieletto, and Emanuele Menegatti. Towards smart robots: Rock-paper-scissors gaming versus human players. In *PAI@ AI* IA*, pages 89–95. Citeseer, 2013.

[41] Amnon Rapoport and David V Budescu. Generation of random series in two-person strictly competitive games. *Journal of Experimental Psychology: General*, 121(3):352, 1992.

[42] Tobias Reichenbach, Mauro Mobilia, and Erwin Frey. Mobility promotes and jeopardizes biodiversity in rock–paper–scissors games. *Nature*, 448(7157): 1046–1049, 2007.

[43] Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. In *Recommender systems handbook*, pages 1–35. Springer, 2011.

[44] Nicolas Riesterer. Variational conceptor learning for generative modelling of sequential data. Master's thesis, Albert-Ludwigs-Universität Freiburg, 2018.

[45] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach (International Edition)*. {Pearson US Imports & PHIPEs}, third edition edition, 2010. ISBN 978-0-13-207148.

[46] Barry Sinervo and Curt M Lively. The rock-paper-scissors game and the evolution of alternative male strategies. *Nature*, 380(6571):240, 1996.

[47] Vernon L Smith. An experimental study of competitive market behavior. *Journal of political economy*, 70(2):111–137, 1962.

[48] Vernon L Smith. Corporate financial theory under uncertainty. *The Quarterly Journal of Economics*, 84(3):451–471, 1970.

[49] A. P. Sutiono, R. Ramadan, P. Jarukasetporn, J. Takeuchi, A. Purwarianti, and H. Iida. A mathematical model of game refinement and its applications to sports games. *EAI Endorsed Transactions on Creative Technologies*, 15:1–7, 2015.

[50] Carol Vogel. Rock, paper, payoff: Child's play wins auction house an art sale. 04 2005. URL `http://www.nytimes.com/2005/04/29/arts/design/rock-paper-payoff-childs-play-wins-auction-house-an-art-sale.html`.

[51] Willem A Wagenaar. Generation of random sequences by human subjects: A critical survey of literature. *Psychological Bulletin*, 77(1):65, 1972.

[52] Zhijian Wang, Bin Xu, and Hai-Jun Zhou. Social cycling and conditional responses in the rock-paper-scissors game. *arXiv preprint arXiv:1404.5199*, 2014.

[53] Claus Wedekind and Manfred Milinski. Human cooperation in the simultaneous and the alternating prisoner's dilemma: Pavlov versus generous tit-for-tat. *Proceedings of the National Academy of Sciences*, 93(7):2686–2689, 1996.

[54] David Weibel, Bartholomäus Wissmath, Stephan Habegger, Yves Steiner, and Rudolf Groner. Playing online games against computer-vs. human-controlled opponents: Effects on presence, flow, and enjoyment. *Computers in Human Behavior*, 24(5):2274–2291, 2008.