Otto von Guericke University Magdeburg

Master Thesis

Prediction of Player Moves in Collectible Card Games

Author:
Tony Schwensfeier

Examiner:
Prof. Dr. habil. Rudolf Kruse

Supervisor:
M.Sc. Alexander Dockhorn

A thesis submitted in fulfillment of the requirements
for the degree of Master of Science

in the

Chair of Computational Intelligence
Institute for Intelligent Cooperating Systems

OTTO VON GUERICKE
UNIVERSITÄT
MAGDEBURG

INF

FAKULTÄT FÜR
INFORMATIK

March 11, 2019

STATEMENT OF AUTHORSHIP

Thesis:

Name: Tony Surname: Schwensfeier

Date of Birth: 21.09.1993 Matriculation no.: 200068

I herewith assure that I wrote the present thesis independently, that the thesis has not been partially or fully submitted as graded academic work and that I have used no other means than the ones indicated. I have indicated all parts of the work in which sources are used according to their wording or to their meaning.

I am aware of the fact that violations of copyright can lead to injunctive relief and claims for damages of the author as well as a penalty by the law enforcement agency.

Magdeburg, 11.03.2019

Signature

Abstract

Prediction of Player Moves in Collectible Card Games

The prediction of moves of opposing players in a game is a task of each CI which wants to call itself intelligent. In Collectible Card Games (CCG), this challenge is complicated by a great number of strategies, or decks, and, therefore, the great set of moves, or cards, the opponent can choose from. So how can a CI method be accomplished, which maximizes the number of correct card predictions in a CCG? During this thesis, a CI method was developed which can predict cards and decks based on a data set of recently played decks. These decks defined the present metagame of the game 'Hearthstone'. Card were predicted by stochastic calculations on the hierarchically clustered decks. In the process, different distance functions and linkage methods for the hierarchical clustering were evaluated. A 90% correct card prediction on card sequences of decks exclusively from the metagame and a 40% correct card prediction on card sequences of decks evolving from the metagame were achieved which was an increase of 20% in contrast to the purely stochastic based prediction method.

Acknowledgements

At this place, I want to acknowledge my examiner Prof. Dr. habil Rudolf Kruse and my supervisor Alexander Dockhorn who made possible to work at this thesis. I want to thank my supervisor for the time that he offered when I had problems or needed advice, and especially for his guidance in plotting data which I gained during my evaluation.

Contents

Chapter 1

Introduction

Enthusiasm for Computational Intelligence (CI) is part of human history since the early days of the first automatons. This enthusiasm was routed in the wish to understand human nature and to improve the life through this new knowledge. The earliest ideas of realizing Computational Intelligence were automatons. After the first attempts of developing such machines were exposed as a swindle, where people controlling the machine from the inside, genuinely autonomous automatons were created. Examples are the writing and drawing automatons of the father and his son Jaquet-Droz around 1770. Multiple decades, their machines could be visited as they traveled through Europe.[1]

With the early understanding of the human brain through the newly established science of neurology in the years of 1930 and 1940, approaches to realize an Artificial Intelligence (AI) were developed. Important foundations were made by Alan Turing's theory of computation which defined the theoretical functionality of a computer, based on '0' and '1'. Furthermore, it was possible with his work to design the Turing test which claimed that the consideration of a machine as 'intelligent' is linked to the missing distinguishability of its response as human or mechanical.[2][3]

The invention of the computer as we know in the years of 1950 and its predecessors inspired people to the development of AI. The first official founding of this discipline was in 1956. In the center were the game checkers. Nevertheless, the agents, AI-based software programs, in these games were not able to challenge the best players for a long time. They were based on simple brute-force techniques which, despite their simplicity, created good results. As a consequence, most of the research focused on the improvement of brute-force performance in developing faster data structures and search algorithms.

In the following decades, many significant breakthroughs were made. The first one was the improvement of the computational processing power and the second one was the development of new techniques, based on the realization that brute-force can not be improved further. To that time, after many iterations, IBM's agent Deep Blue was able to defeat the world champion in chess in 1996. [4] Furthermore, the term 'Computational Intelligence' was introduced which used in this work instead of referencing to the term 'AI'.

Why is CI still relevant? The usage of Computational Intelligence, as software on a home computer, robot or other devices has some significant advantages over human work. CI does not get tired, has few needs, remembers more efficient and works faster than humans. There are still areas, CI can not faultlessly be used or can not be used at all, like government or emotional related work. They have still problems to understand the new tasks, which they are not developed for. Yet there is the opportunity that they will do the kind of work, a human can not do or are not willing to do. I think that we should continue to develop CI further so that it can be used in further areas. Recent developments highlight some interesting applications for CI in society.

First, In the industrialized nations, we stand in front of the problem of an aging population. Robots can become our caretakers and companions. Examples are the robot caretakers for lifting elders, Pepper the informational robot or Paro, the socially assistive seal robot. [5, 6, 7]

Second, the high signal transmission time between mars and earth up to 22.3 minutes makes it impossible to operate a Mars Rover from Earth in real time. Furthermore, it is currently not possible to send an operator to mars. So a software, named AEGIS, is used to determine the next target of the latest Mars Rover, Curiosity. [8, 9]

Third, a CI which is able to win against world champions in Chess, Shogi, and Go, points out the potential from CI as opponents and teachers. So multiple shogi players in Japan use teaching software to increase their skill. [10]

1.1 Games

Computational Intelligence finds multiple usages in games, in their development, debugging and in the game as opponent or administrative system. Characteristics of a game in general, particularly Trading Card Games, and the problems which they create for CI, are covered in the following. Possible solutions are discovered by using the game "Hearthstone" in this thesis later on. Basics of this game are explained in this section.

1.1.1 What is a Game?

From the scientific point of view, a game follows a sequence of moves from a given starting state. The state of a game is called game state. The moves translate the game in from one game state into another. They are chosen by players among possibilities or determined randomly by something like a dice roll or mixing cards. A different amount of knowledge is offered to each player about the game, and when the game ends or during the game, players get rewards. [11]

So first, we distinguish based on the outcome of moves with regard to their randomness:

- In deterministic games, the outcome of a move is determined before it is made, like chess, where the next position of a piece is determined by its current position and the player's move.

- In stochastic games, the outcome of moves is based on stochastic events. Game examples are poker, roulette or Yahtzee, where the next game state is determined by the stochastic result of shuffling cards, spinning a wheel or throwing dice.

A game can have deterministic elements as well as stochastic elements.

Second, we distinguish a game by the amount of available knowledge:

- With perfect information, each player has the complete knowledge of the state of the game and every player has the same information. Examples are games like chess or checkers, in which all pieces on the board are visible to all players

- With partial(imperfect) information, each player has the knowledge of a part of the game state. Examples of such games are poker or bridge. In poker, the revealed cards are visible to all players. Nevertheless, each player only knows the cards of its hand and has no complete knowledge of the cards in the deck and their order.

Third, a game is characterized by its reward. It is represented in the form of ordinal data, win, draw and loose, or in the form of numerical data, as the number of chips won in a poker game.

1.1.2 Card Games

Card Games are a prevalent type of games, and they generally have partial information and the outcome is strongly dependent on stochastic events. These characteristics have a fascinating influence on a working CI, which can not calculate the next possible moves with the limited amount of information of the game state, so it has to guess about the state.

Each card game consists of one or multiple decks. A deck is a set of cards, which, in classic card games, depends on mostly unique cards with common themes. In the case of French playing cards, there are four kinds of suits, numbered and faced cards. Each Player gets a number of randomized cards before the game begins and/or during the game, which are not known by the other players. Nevertheless, the players know the deck and can guess the cards of their opponent by calculating based on knowledge of their cards and the already played cards. They try to maximize their reward over time while playing. Not knowing the cards of the opponent, yet knowing the cards of the deck reduces the number of possible cards. Another type of card games removes this knowledge base, replaces it with unique decks and thereby creates an imperfect information game game with less information. Therefore, other methods have to be used to guess the opponents' cards. This type of card games is described in the following.

FIGURE 1.1: Logos of popular CCGs, from left to right: Hearthstone[15], MTG[12], Pokémon TCG[13], Yu-Gi-Oh![14], Shadowverse[16]

1.1.3 Collectible Card Games

In 1993 a new type of card game was released by the company 'Wizards of the Coast'[12]. It was a Collectible Card Game (CCG) called 'Magic the Gathering', in short, 'MTG'. Later on, driven by the big financial success of the game, other companies made their own game, popular examples are 'Pokémon: Trading Card Game', in short, 'Pokémon TCG'[13], and 'Yu-Gi-Oh!'[14]. The logos of some of the current most popular CCGs are shown in Figure 1.1.

In CCG, each player creates its deck, based on deck-building rules, using a set of cards, which were produced for this game. Players can purchase these cards by buying booster packs, a packed set of randomized cards. The cards have different types and, therefore, follow a different role in the game. As resource cards, they pay for other card types, like creature cards, which can attack, or spell and object cards, which can have multiple purposes. The players try to achieve a goal, which lies, in most cases, in reducing the number of points of the opponent to zero or destroy a certain number of creatures.

A good built deck is a good strategy and the key to win the game. However, since cards are randomly distributed, they are hard to collect. Because of this reason, players trade their cards with one another. In consequence, they can also be found by the name 'Trading Card Game'(TCG). Nowadays, the name is less applicable, because of the recently developed online CCG, which do not allow this kind of interaction between players anymore, yet offer another way to collect the needed cards.

1.1.4 Hearthstone

One of the most popular online CCGs is 'Hearthstone: Heroes of Warcraft'[15], in short, 'Hearthstone', which was released in 2014 and developed by the company 'Blizzard Entertainment'. Its popularity created a big community which collects huge amounts of information about the games. The active community created APIs which results in the growing interest of scientists to use 'Hearthstone' as a test platform for their AIs. The following schematic overview of the game rules will be important during this thesis. The more specific rule set can be accessed on *Hearthstone Wiki* page [17]. For other related work to this subject, read more in 2.5.

In this CCG, a deck is built around one out of nine possible heroes, which represent one player in a game. They have attributes like health, which can be reduced by taking damage and can sometimes even attack the opponent. If the health of a player's hero reached zero the hero dies and the player loses the game. The health can

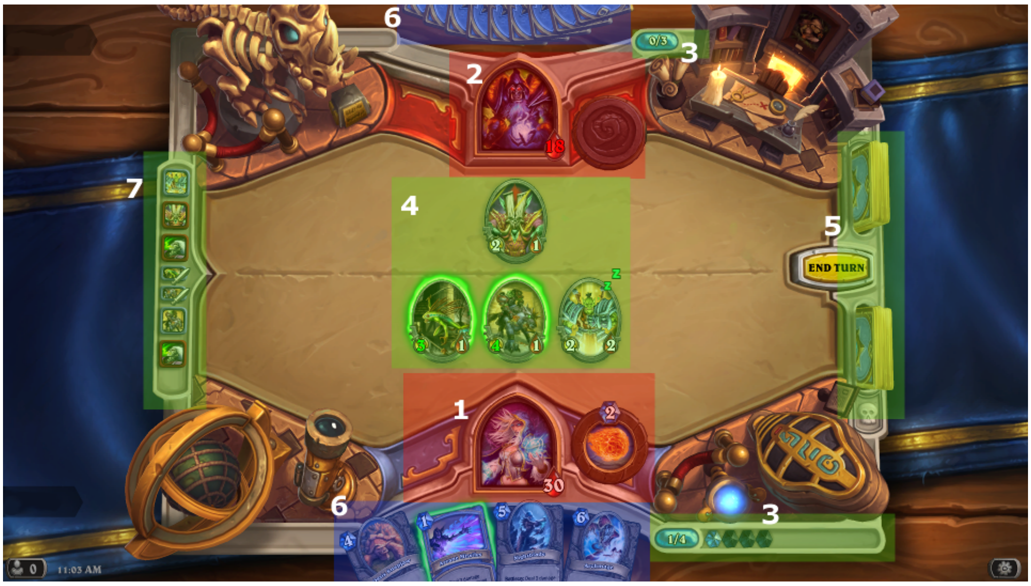

FIGURE 1.2: Example of a Hearthstone game board, 1. Player and its Ability, 2. Opponent and its Ability(already used), 3. Mana, 4. Opponent's minions (top row) and player's minions on the board(bottom row), 5. Decks, 6. Hand cards, 7. Play history

be protected by additional armor for the hero, who receives it through card effects. Destroying the hero of the opponent is the goal of the game. Furthermore, heroes have an ability, which can be activated once each turn. The game is separated in alternating turns, where players draw a card from their deck, play cards and attack with their creatures. They begin the game with zero Mana, which increases by one in the player's turn until it reaches a maximum of ten Mana. The deck of each player contains 30 cards that the player can choose out of a set of over thousands of cards. They are either free or restricted to one specific hero. The game format the deck is played in is an additional restriction. Hearthstone provides two game formats.

- The **Standard** format is a balanced format, which permits cards of card sets of the current or previous calendar year, as well as basic sets. Sets of new cards are added to the game several times a year.
- The **Wild** format is unrestricted, i.e., every card released can be played in it.

Both formats can be played in 'Ranked' and 'Casual Play Mode', whereby the performance of a player and the number of games played in the 'Ranked Play Mode' defines the player's 'Rank' which ranges from the lowest rank fifty to the highest rank zero.

The cards in Hearthstone have five grades of rarity, which is a measure to represent their general strength. Free, common, rare and epic cards are allowed to be contained twice in a deck, legendary cards are allowed once. There are four main card types. They are shown in Figure 1.3.

- **Minion cards** are the creatures in Hearthstone. They can be played by paying their Mana costs and are permanently in play until they are destroyed. They are positioned in a single line for each player which contains up to 7 minions. Minions have an attack value and a health value. Minions can either attack minions of the opponent or the opponent's hero. When minions attack one another, each takes the damage in the amount of the others attack value, which decreases their health by this value. When it attacks a player, the player's hero health is decreased by this value. The health does not regenerate over time, and a creature dies if its health value reaches zero. Minions can have triggered abilities, which activates when played, when dying and other situations, or passive abilities, such as preventing the opponent's creatures in attacking the player.

- **Spell cards** can be played by paying their Mana costs and leave the play after their ability was processed.
- **Weapon cards** can be played by paying their Mana costs. They have an attack value and durability value. They enable the hero of a player to attack minions or the other player. By doing so, the hero does damage equal to the attack value of the weapon. In return, the hero will receive damage in the amount of the target's attack value, and the durability of the weapon is reduced by one. A weapon will be destroyed if its durability reaches zero or when a new weapon is played.
- **Hero cards** can be played by paying their Mana costs. When played, they replace the current hero and its ability. They keep the hero's attack, yet add armor to its health.

When cards interact with each other, it is called a synergy which might be stronger than an ability by itself. Such as a minion card which regenerates the hero's health whenever a minion is played and a second minion card which damages the opponent hero's health whenever the own hero regenerates health. Cards can have a value for the game, on their own or in synergy with other cards. So building a deck requires specific knowledge of the cards. Deck building is a factor that influences the chances of winning a matchup before the game started. Understanding the cards, the types of deck they form, and the advantages of decks against others results in interaction between the players. They choose promising decks and develop new types of decks which might have new advantages. That creates a game in itself, which is called metagame.

1.2 Research Questions

Problems which arise in playing CCG can be summarized as the following. Winning the game requires (I) to optimize the own strategy and on further to (II) uncover the strategy of the opponent player, so the moves of the opponent can be predicted and countered. Both strategies influence the (III) chance of winning. Uncovering the

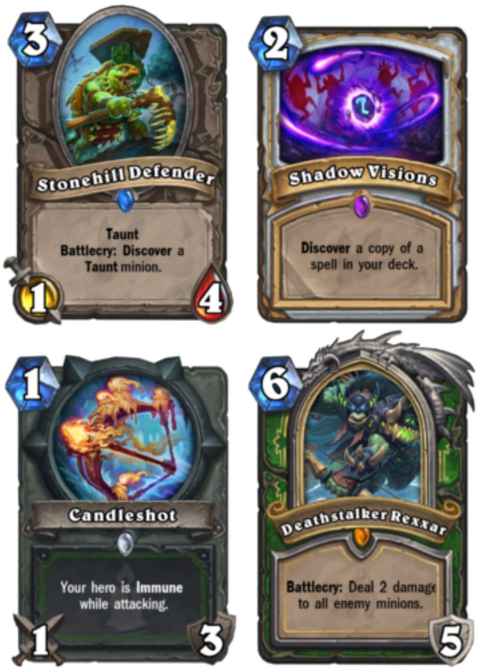
3
Stonehill Defender
Taunt
Battlecry: Discover a Taunt minion.
1
4
2
Shadow Visions
Discover a copy of a spell in your deck.
1
Candleshot
Your hero is Immune while attacking.
1
3
6
Deathstalker Rexxar
Battlecry: Deal 2 damage to all enemy minions.
5

FIGURE 1.3: Example of the different card type, from top left to bottom right: minion card, spell card, Weapon card, hero card[18]

opponent's strategy is decisive influenced by the understanding of the metagame which evolves by the creation of optimized strategies. That leads to the question.

How can a CI method be accomplished, which maximizes correct card predictions using the metagame?

This research question can be specified by subdividing it into sub-questions which this thesis tries to answer.

1. How can the metagame be represented?
2. Which information are derivable from the metagame?
3. What can be deduced from this information to predict cards?

1.3 Structure of this thesis

In the next chapter, the background knowledge is discussed which is build upon to answer the questions, this thesis wants to answer. Related work shows related problems and how they were solved.

Chapter 3 covers the problems which may occur and has to taken into account when working with Hearthstone as a test environment and meta data. Different solution strategies are discussed and proposed to cluster the meta data, and predict the opponent's deck and cards.

Multiple implementation variances for the clustering are considered and evaluated in chapter 4. The card prediction algorithm is evaluated by its correct predictions.

In the finishing chapter, the results are presented and interpreted. Various concepts are suggested to extend the thesis and the use of the result in other areas than Hearthstone and games.

Chapter 2

Background

This chapter discusses some background, the thesis is build around. First, some general topics on game playing in section 2.1 and machine learning in section 2.2 are considered. Next, basics of stochastic are in the focus in section 2.3 and (fuzzy) multisets are explained in section 2.4, which this thesis is build up on. Finally, some important previous work is summarized in section 2.5, that has been published on Card Games, especially CCGs like 'Hearthstone', and machine learning.

2.1 Game Theory

In this section, we will discuss the core concepts of Game Theory. Their influence on CCG is discussed in section 3.1 later on. Game Theory is by *K. Salen*, *E. Zimmermann* [11], and *J. Schell* [19].

In game theory, a game is distinguished by its characteristics. A game has a combination of the following characteristics

Cooperativeness: In a cooperative game, players are able to make agreements, which will have a positive outcome for both sides. On the other hand, a game is non-cooperative, if players can not form alliances, this may be enforced by the game rules itself or the decision of the players, or that the outcome of a game is not positive for all players.

Symmetry: In a symmetric game the payoff is based on the strategies combination and not the players. Exchanging the players' results in the same payoffs. Table 2.1 shows the payoffs of a symmetric strategy and table 2.2 of an asymmetric strategy.

Zero-Sum: Is the Sum of the payoffs of all player for all strategy combinations equal 0, it is a zero-sum game. The strategies neither decrease nor increases the resources of the game. Table 2.3 shows the payoffs of an zero-sum game and table 2.4 of a non-zero-sum game.

	A	B
A	0, 0	-2, 1
B	1, -2	-2, -2

TABLE 2.1: symmetric strategy

	A	B
A	-2, -2	1, -2
B	1, -2	0, 0

TABLE 2.2: asymmetric strategy

	A	B
A	0, 0	-2, 2
B	2, -2	-4, 4

TABLE 2.3: zero-sum game

In context to the symmetry and zero-sum is the **Nash equilibrium** which describes a situation in which two players keep their strategy, since the change of the strategy results in a worse payoff for the changing player. The strategies A(row) and B(column) of Table 2.3 shows such a situation.

Move-Order: The Move-Order can be one of these two. First, is the action-order simultaneous, both players make their move at the same time or in the unawareness of the other players' previous move. Second, is the action-order sequential, players know previous players' actions.

Information: Perfect information games are a subset of sequential games, where all players have complete information on all the other players' previous moves, for example, chess and Go. In most games, players have imperfect information(partial information) of the game state, like Poker.

Game length: We can distinguish games by their length, the maximum number of total possible game steps, in which players make their move. Some games consist of only one game step, like Rock–paper–scissors. Other games have multiple or an infinite number of game steps, which is the case for many single player or cooperative multi-player video games.

Metagame: It is summarized in a video of *Stanislav Costiuc* [20] The metagame ('meta' from Greek for 'about', 'beyond') is a literally 'a game beyond the game'. Depending on the context, the word is used two describe two different things. The **mechanical metagame** is a set of systems and mechanics which is wrapped around the gameplay by the developer. The **emergent metagame** is strategies that transcend the rules and systems of a game which emerging from the player interactions. This definition of the metagame is used in game theory and which is used in this thesis further on.

The metagame occurs specifically in competitive games is strategies the players utilize. Considering a Rock–paper–scissors game, the strategy of a player is not only based on the payoffs of the current game, but also on the meta factors of the previous game. If the opponent takes every time the move 'rock' the move 'paper' premises a better payoff in the metagame. Though this scenario might be promising for the current state of the metagame, it is not for the following games, since the metagame is emergent. It means that the opponent will adjust his strategy to the metagame

	A	B
A	-2, -2	1, -2
B	1, -2	0, 0

TABLE 2.4: non-zero-sum game

which results in a cycle of the metagame. The emerging of strategies is very specific to CCGs. It can be divided into four phases:

1. In the **experimentation** phase, many strategies are tried out by the players. No strategy is dominant, i.e., is played signally played more than others.
2. In the **stabilization** phase, effective strategies which have been found during the experimentation become dominant. Strategies against strategies and moves to defend the own strategy are found. It is a phase of balancing.
3. In the **meta solved** phase, all strategies are known, resulting in a few dominant strategies and the payoff of a game becoming mainly a matter of player skills.
4. In the **shake up** phase, a new strong strategy is found, a balance is released which changes the rules, or a new expansion with new rules and moves is released, resulting in new experimentation or stabilization which length is depended on the impact of the shake up.

2.2 Machine Learning

Machine Learning are AI Methods which focus on building a mathematical model of sample data, also training data, for the purpose of making decisions or predictions without being told to do so. Types of machine learning are supervised and semi-supervised learning, unsupervised learning and reinforcement learning. Supervised learning and especially unsupervised learning are covered in the Data Mining book [21] and in the book by *Togelius* and *Yannakakis* [22] and explained in the following.

2.2.1 Supervised Learning

Supervised learning is a process to train a CI agent with labeled data to find characteristics in the data the label can be identified with. To do so, it approximates the underlying function between labeled data and their corresponding attributes and features. For instance, to distinguish between knife and fork, an agent receives a set of data to learn from, examples of cutlery, which contain the information about their attributes, like form, size and material, and their corresponding labels (target output), such as fork and knife. So it is supervised because the target output is known. After learning is complete, the algorithm should be able to tell if new unseen cutlery is a knife or fork by its attributes. The goal is, to learn from in-and-output pairs and to create a mapping function which is able to work with new unseen instances of input data.[22] Three main types of supervised learning algorithms can be identified by the data type of the labels:

Classification is an attempt to predict categorical class labels(discrete or nominal) such as knives and forks.

With **regression** it is possible to predict intervals such as the length of a rain shower.

Preference learning teaches a CI to rank new data and to give ordinal output, such as ranking a social media post by its obscenity or racism.

2.2.2 Unsupervised Learning - Clustering

Unsupervised Learning is a process to find and analyze similarities between data input like supervised learning, except the learning data are not labeled. In other words, the target output is unknown. The algorithm searches for patterns in the input data attributes. The focus is on intrinsic structures of the input data and in their associations instead of predicting target values.[22]

One subfield of unsupervised learning is clustering. The task of clustering is to find groups (cluster) in a number of data points so that the attributes of each point in a group are similar and dissimilar to attributes of other groups' data points. It is used for group detection in multidimensional data and reducing tasks, such as data compression, noise smoothing, and outlier detection. Since the target clusters are unknown, good clusters are characterized by high intra-cluster similarity, or high compactness, and low inter-clusters similarity, or good separation. However, these properties are no measure of a cluster's meaningfulness.

Furthermore to the characteristics described above, cluster algorithms are defined by their membership function and a search procedure. The membership function defines the degree of assignment from the data samples to a cluster. The search function describes the strategy of clustering the data using the membership function, such as separating all data points into clusters at once as in k-means and DBSCAN, or recursively as in hierarchical clustering. These three clustering procedures are explained in the following.

2.2.2.1 k-means clustering

k-means is a partitional-clustering algorithm and a vector quantization method, which is because of its good balance between simplicity and effectiveness considered the most popular clustering algorithm. It follows a data partitioning approach of partitioning data samples into k clusters, so that the sum of distances (or quantization error) between data points and their corresponding cluster centers is minimized.[22]

A cluster is defined by its data points and its cluster center, which is the average of all to the cluster assigned data points. Each data point is assigned to the cluster with the smallest distance. The clustering algorithm requires multiple iterations of alternating between calculation of the cluster center and reassigning of the data points to the closest cluster center, begging with a randomized allocation of data points to clusters which number is defined beforehand. The basic steps are the following:

Given k

1. Randomized partitioning of data points into k non-empty clusters
2. Calculation of the cluster centers
3. Assignment of each data point to the closest cluster center
4. Stop if each data point remains by it current cluster center, else continue with 2

Data points are objects which can be described as a vector. Depending on the data set, it is beneficial to use different ways to calculate distances of two vectors $p = (p_1, p_2, \ldots, p_n)$ and $q = (q_1, q_2, \ldots, q_n)$. Three distance functions are:

- The **Manhattan distance** is sum of distances of two vectors p and q on each axis.[23]
$$dis_{Manh}(p,q) = \sum_{i=1}^{n} |p_i - q_i| \tag{2.1}$$
- The **Euclidean distance** is the square root of the sum of squared distances of two vectors p and q on each axis.[23]
$$dis_{Eucl}(p,q) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} \tag{2.2}$$
- If the data points are sets A, B which are not associated with a vector, the **Jaccard index** offers a way to measure the similarity and diversity of the sets. By its subtraction from 1, the **Jaccard distance** is calculated.[24]
$$dis_{Jac}(A,B) = 1 - J(A,B) = 1 - \frac{|A \bigcap B|}{|A \bigcup B|} \tag{2.3}$$

Despite its effectiveness, k-means has some disadvantages and is not applicable in some situations. First, data objects have to be in continuous space. Second, it only finds convex clusters what results in favoring hyper-spherical clusters. Third, the number of clusters k has to be specified in advance. Finally, k-means is weak against outliers, data points with extremely high distances to others, since they may distort the distribution of data and may influence the performance of the algorithm. Hierarchical clustering is a different approach of clustering which fixed some of these issues and is described in the next paragraph.

2.2.2.2 Hierarchical clustering

Hierarchical clustering is a clustering method which attempts to build a hierarchy of clusters. Two methods exist to build a hierarchy. The agglomerative strategy is an bottom-up method which gradually merges clusters together until a termination criteria is met or one single cluster is left. In contrast, the divisive strategy creates cluster hierarchies by splitting up clusters top-down until every data point is in its own cluster, starting with a single cluster which contains all data points. Both using distance matrices as cluster strategy.[22]
basic steps of an agglomerative clustering algorithm is presented in the following:

Given k

1. Creation of one cluster for each data point
2. Creation of the distance between, containing the distance of all clusters
3. Finding and merging of the closest clusters
4. Stop if there are k clusters; otherwise continue with step 2

The distance between is represented by a distance matrix. The calculation of the distances matrix, and the merging of clusters requires linkage methods which use distance functions, such as Euclidean or Manhatten distance. The following linkage methods can be distinguished:

- **single linkage**: This method searches for the closest points between two clusters(or inter-cluster distance), their distance is the cluster distance. Once the distance between all clusters are calculated. The clusters with the smallest distance are merged. The feature of single linkage is, that it leads to chains in the data.[25]
- **complete linkage**: In contradiction to single linkage, this method tries to find the biggest inter-cluster distance, i.e. the farthest points between to clusters, their distance is the cluster distance. As in single linkage the closest clusters are merged. Complete linkage and the following methods tend to create circular clusters.[25]
- **average linkage**: The cluster distance in this method is the average over all pairwise distances between all data points between two clusters. As before, clusters with the closest distance are merged.[25]
- **centroids**: In contrast to average linkage, this methods calculates the average of all data points and creates a centroid for each cluster before measuring the distance. The clusters with the closest distance of their centroids are merged.[25]
- **ward's minimum variance method**: Similarly to centroids, centroids are created of each cluster. Next, a centroid for a simulated merge of both clusters is created. The distances of data points to their centroid are compered to the

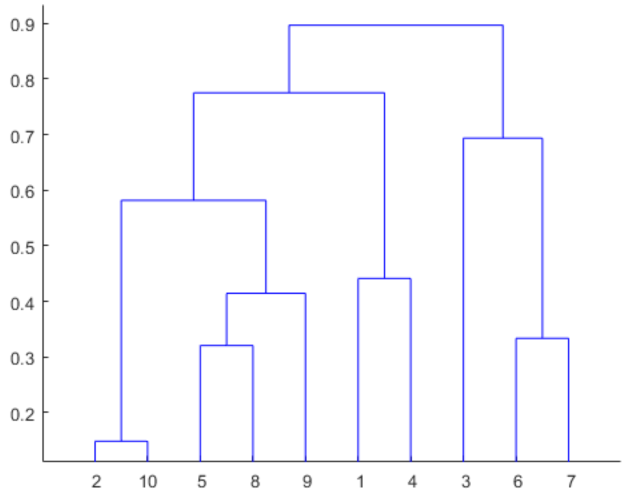

FIGURE 2.1: Dendrogram plot example [27]

distance of them to the simulated centroid of both clusters. Clusters with the smallest deviation are merged. [26]

Once all cluster steps (merging or splitting) are completed, each step of the algorithm can be visualized in a tree representation. With the cluster which contains all data points as root, each merging clusters as children merged clusters as their parent and one-elemental cluster in the lowest layer. The edges of this, so-called dendrogram, are weighted with the merging (or splitting) costs, in this case, the cluster distance. The final cluster is created by cutting the dendrogram at the desired level of merging costs. A plot example of a dendrogram can be seen in Figure 2.1.

Hierarchical clustering is greedy, meaning that it merges local optima, which may not result in the global optimum.

2.2.2.3 DBSCAN

In focus of density based clustering is the concept of finding dense regions of data points with low density space (noise) between them. The main representative of density based clustering is DBSCAN. It detects clusters by the high density of data points inside the cluster and low density outside of it. The clustering process is incremental.[21]

DBSCAN is based on the two concepts of **density reachability** and **density connectivity**. Both build upon two input variables, the size of the epsilon neighborhood (ϵ) and the minimum points in range (m). DBSCAN has the key idea that, for each point of a cluster, at least m other points are in the neighborhood of range ϵ.

Density reachability ask for the question, whether two points are in the same cluster. A point p_1 is **direct density reachable** from p_2 if the distance of p_1 and p_2 is less than ϵ, and at least m points are in ϵ-neighborhood of p_2. A point p_n is density reachable from a point p_0 if a sequence of points between them exists whose points are direct density reachable from the previous one beginning by p_0. Density connectivity is the build-up step of DBSCAN. If two points p_1 and p_2 are density reachable from a third point p_0, p_1 and p_2 are **density connected**.

DBSCAN classifies points into point type which are *core points*, *border points* and *noise points*. *core points* have at least m points in their ϵ-neighborhood. *boarder points* are in range of a *core point*, though, have fewer than m points in their ϵ-neighborhood. *noise point* is any that that is not *core point* or *boarder point*. The following steps are the main ones of DBSCAN.

1. Selection of a starting point p

2. Retrieval of the point in ϵ-neighborhood of p and classification into point type

3. If the point is a *core point*, creation of a new cluster or extension of exiting cluster or merging of clusters if p is *density reachable* from points in ϵ-neighborhood and points are in different clusters

4. Stop if all points have been processed; else continue with step 2 with the next point in the database

Advantages of DBSCAN include that the number of clusters is not required apriori. Furthermore, it is able to find find arbitrarily shaped clusters and can detect outliers. Since the variable m is selected globally, it may have problems clusters of different densities, though, solutions approaches are existing. Since the algorithm process uses distance operations, it may find the curse of dimensionality problem for high-dimensional data sets.

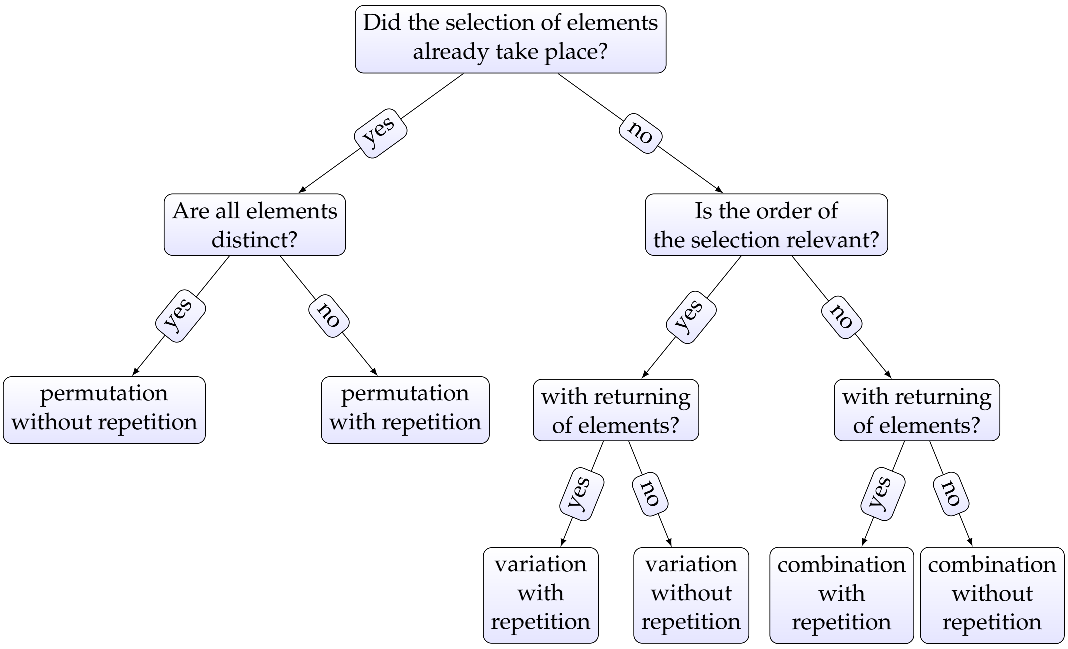
Did the selection of elements already take place?
yes
no
Are all elements distinct?
Is the order of the selection relevant?
yes
no
yes
no
permutation without repetition
permutation with repetition
with returning of elements?
with returning of elements?
yes
no
yes
no
variation with repetition
variation without repetition
combination with repetition
combination without repetition

FIGURE 2.2: Differentiation of permutation, variation and combination with and without repetition

2.3 Stochastic

Stochastic is a sub-field of mathematics which is subdivided into probability theory and mathematical statistics. It is about the number of events, their possible outcome, and their probability and is discussed by *H. Rinne* [28], and *D. Maintrup* and *S. Schäffler* [29].

The **combinatorial analysis** is part of probability theory. Its focus is the consideration of the number of element combinations. The combinatorial analysis can be distributed into three different subfields, which are the permutation, the variation, and the repetition. The distribution is determined by the selection of the elements, the differentiation of the elements, the relevance of the order of the elements and the return of the elements as shown in Figure 2.2.

2.3.1 Permutation

Permutation describes an arrangement of elements. All elements of a set are considered. They may occur once or multiple times. If all element are distinct, i.e. each element occurs once, the permutation is a permutation without repetition. It is defined that each unambiguous image Φ^n of an ordered set $\{1, 2, ..., n\}$ on a set A of size n is a permutation without repetition. $P(n)$ is the number of permutations of n distinct elements.

$$P(n) = 1 \cdot 2 \cdot ... \cdot (n-1) \cdot n =: n! \tag{2.4}$$

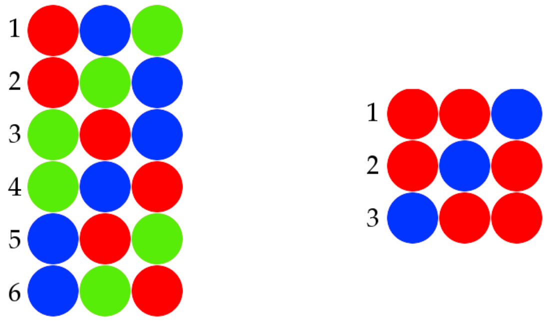

FIGURE 2.3: permutation without repetition(left) and permutation with repetition(right)

If at least an element cannot be distinguished from another one, i.e., it occurs multiple times, the permutation is a permutation with repetition. Then, the matching elements can be switched at their position in the distribution without changing the arrangement of elements. It is resulting in a reduction in the number of possible arrangements. For a permutation with repetition, it applies that when $A = \{a_1, a_2, \ldots, a_k\}$ is a set of k elements and $n_1, n_2, ..., n_k$ in $\mathbb{N}_0$ with $\sum_{i=1}^{k} n_i = n$. Each ordered n-tuple, which contains the element a_j exact n_j times $(1 \leq j \leq k)$, is a permutation with repetition. Then, $P(n|n_1, n_2, ..., n_k)$ is the number of permutations of n elements, where k are distinct.

$$P(n|n_1, n_2, ..., n_k) = \frac{n!}{n_1! \cdot n_2! \cdot ... \cdot n_k!} =: \binom{n}{n_1, n_2, ..., n_k} \tag{2.5}$$

with $\sum_{i=1}^{k} n_i = n$ and $0 \leq n_j \leq n \forall j$.
$\binom{n}{n_1, n_2, ..., n_k}$ is called the polynomial coefficient.

Figure 2.3 shows an example of a permutation with repetition and one without repetition.

2.3.2 Variation

A variation is an ordered sample of elements of a set, i.e. it is an arrangement which, in contrast to the permutation, does not compulsorily contain all elements of the set. If the ordered sample contains all elements of the set, it is a permutation. As in a permutation, the elements can appear once or multiple times. For a variation without repetition, it is defined, that each ordered n-tuple of distinct elements of a set A with the size N is a variation without repetition of N elements of the scope n, with $n \leq N$. A variation without repetition is a sample without returning of elements of scope n from a set of N elements, where the order of the elements is important. Since one element is selected and then removed from the set of remaining selectable elements, the number of elements for the next selection is reduced by one. $V(N, n)$ it the number of distinct variations without repetition.

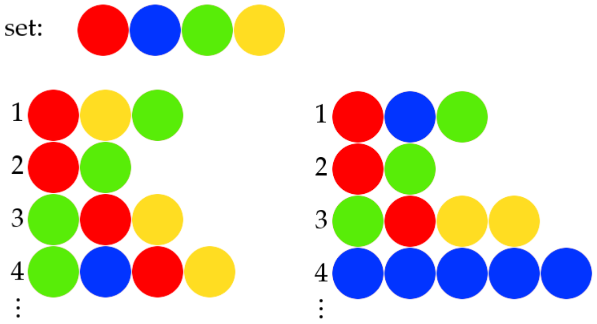

FIGURE 2.4: variation without repetition(left) and variation with repetition(right)

$$V(N,n) = \frac{N!}{(N-n)!} = \frac{P(N)}{P(N-n)} \tag{2.6}$$

In contrast to a permutation, the occurrence of multiple elements in a variation, i.e., the repetition, is created through selection from and returning of elements into the selection pool, instead of the multiple existences of the same element. Each ordered n-tuple of elements of a set A with size N is a variation with repetition of N elements of scope n, with $n \leq N$ or $n > N$. A variation with repetition is a sample with returning of elements of scope n from a set of N elements, where the order of the elements is important. Since elements are returned into the scope after the selection, the number of elements in the arrangement can extend the number of elements in the set. The Number $V^*(N,n)$ of distinct variations without repetition is

$$V^*(N,n) = N^n \tag{2.7}$$

Figure 2.4 shows an example of a variation with repetition and one without repetition.

2.3.3 Combination

A combination is an unordered sample of elements of a set, i.e., it is an arrangement which, as in a variation, does not compulsorily contain all elements of the set, and in contrast to it, does not consider the order of elements. It is resulting in a reduction of the number of possible arrangements, in contrast to the variation. Elements can be selected once or multiple times. Each subset of size n from a set A of size N is a combination without repetition of N elements of scope n, with $n \leq N$. A combination without repetition is a sample without returning of elements, where the order of the elements is unimportant. $C(N,n)$ is the number of distinct combinations without repetition which is calculated by the binomial coefficient $\binom{N}{n}$.

$$C(N,n) = \frac{V(N,n)}{P(n)} = \frac{N!}{(n!(N-n)!} =: \binom{N}{n} \tag{2.8}$$

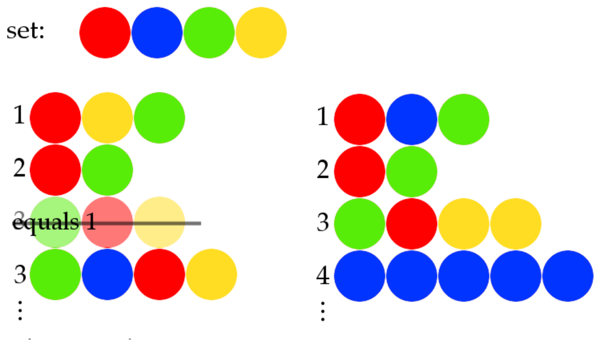

FIGURE 2.5: combination without repetition(left) and combination with repetition(right)

The combination of all variations of N Elements with scope n with returning of elements, which contain the same elements in an equal number, in an equivalence class is a combination with repetition. A combination with repetition is a sample with returning of elements, where the order of elements is unimportant. The number $C^*(N,n)$ of distinct combinations with repetition is

$$C^*(N,n) = \binom{N+n-1}{n} \tag{2.9}$$

Figure 2.5 shows an example of a combination with repetition and one without repetition. It is visualized that two samples in different order of elements are the same, since the order is unimportant.

2.3.4 Experiments and events

An **experiment** is a repeatable process with an uncertain result under approximately equal conditions, such as throwing dice. One possible result is called its **outcome** ω_i with $i = 1, 2, \ldots, N$ and the **sample space** Ω is the set of all distinguish outcomes. In case of six-sided dice, the sample space contains the eye numbers one to six.

An **event** is a subset of outcomes of an experiment, such as the eye numbers one to three. A **certain event**, i.e., an event that must occur, are the eye numbers one to six since it is equal the sample space. A **random event** $A, B, C, \ldots$ is a strict subset of (omega) and can be the eye numbers one and two. The **elemental event** is a one-elemental random event, such as the eye number one.

For the throwing of a four sided dice, the events are as followed.

- elemental events: $\{1\}, \{2\}, \{3\}, \{4\}$
- other random events: $\{1,2\}, \{1,3\}, \{1,4\}, \{2,3\}, \{2,4\}, \{3,4\}$, $\{1,2,3\}, \{1,2,4\}, \{1,3,4\}, \{2,3,4\}$
- certain event: $\{1,2,3,4\}$

2.3.5 Probability

The probability can be categorized definitely. In this work, it is defined in the prior and objective way. The probability of an event $Pr(A)$ is indicated by the quotient of the size of an event $N(A)$, i.e., the number of outcomes of an event, and the size of the sample space N, i.e., the number of all possible outcomes.

$$Pr(A) = \frac{N(A)}{N} \tag{2.10}$$

The **conditional probability** of A given B P(A | B), given the two events A and B, is defined as:

$$Pr(A|B) = \frac{Pr(A \cap B)}{Pr(B)} \tag{2.11}$$

$Pr(A \cap B)$ is the probability that events A and B occur at the same time. $A \cap B$ is called Intersection. In case of a sample space of men and women with different ages, an intersection would be the selection of a 40 years old man. The event A and B are stochastically independent, if $Pr(A \cap B) = Pr(A) \cdot Pr(B)$. According to the multiplicity theorem, the intersection of multiple events is calculated as follows.

$$Pr(\cap_{i=1}^{n} A_i) = Pr(A_1)Pr(A_2|A_1)Pr(A_3|A_1 \cap A_2)\ldots Pr(A_n|A_1 \cap A_2 \cap \ldots A_{n-1}) \tag{2.12}$$

2.4 Multisets

The notation of multisets are discussed by *S. Miyamoto* [30], extracted from the work of *D. E. Knut* [31]. Multisets are sets which can contain multiple instances of an element. A crisp mulitset M is defined as

$$M = \{k_1/x_1, \ldots, k_n/x_n\} \tag{2.13}$$

over the the basic set $X = \{x_1, \ldots, x_n\}$ with $k_i = \gamma_M(x_1)$ the number of appearances, or count, of x_i in M.

Assume a basic set of elements $X = \{x, y, z, w\}$. For example two objects with the label x, one object with the label y, zero objects with the label z, and three objects with the label w are the multiset $\{x, x, y, w, w, w\}$. It can also be written as $\{2/x, 1/y, 0/z, 3/w\}$ to show the number for each element of the set X, or $\{2/x, 1/y, 3/w\}$ by ignoring elements which are not appear. In the example we have

$$\gamma_M(x) = 2, \gamma_M(y) = 1, \gamma_M(z) = 0, \gamma_M(w) = 3$$

The following are the basic relations and operations for crisp multisets. (inclusion):

$$M \subseteq N \Leftrightarrow \gamma_M(x) \leq \gamma_N(x), \forall x \in X. \tag{2.14}$$

(equality):

$$M = N \Leftrightarrow \gamma_M(x) = \gamma_N(x), \forall x \in X. \tag{2.15}$$

(union):

$$\gamma_{M \cup N}(x) = max\{\gamma_M(x), \gamma_N(x)\} = \gamma_M(x) \vee \gamma_N(x) \tag{2.16}$$

(intersection):

$$\gamma_{M \cap N}(x) = min\{\gamma_M(x), \gamma_N(x)\} = \gamma_M(x) \wedge \gamma_N(x) \tag{2.17}$$

(addition):

$$\gamma_{M \oplus N}(x) = \gamma_M(x) + \gamma_N(x) \tag{2.18}$$

(subtraction):

$$\gamma_{M \ominus N}(x) = 0 \vee \gamma_M(x) - \gamma_N(x) \tag{2.19}$$

The symbols $\vee$ and $\wedge$ are the infix notation of max and min. Consider $N = 1/x, 2/z, 2/w$ and the first example of M, Then,

$$M \cup N = 2/x, 1/y, 2/z, 3/w$$

$$M \cap N = 1/x, 2/w$$

$$M \oplus N = 3x, 1/y, 2/z, 5/w$$

$$M \ominus N = 1/x, 1/y, 1/w$$

Furthermore, an extension of multisets, fuzzy multisets, are featured by *S. Miyamoto* which where firstly discussed by *R. R. YAGER* [32]. While a fuzzy extension is not further discussed here, it was discussed in a recently submitted paper, which builds up on the work of this thesis [33].

2.5 Related work

Previous work to the game Hearthstone show implementation approaches of CI in the game. A central element is the evaluation of game states as discussed in Subsection 2.5.1. From the current game state, a simulation is started to evaluate possible moves by their payoff, though, future game states which are required for the simulation are uncertain. A simulation based on card prediction and game state evaluation is discussed as in Subsection 2.5.2. The approaches can deliver good results as the prediction of cards with bigrams as discussed in Subsections 2.5.3, though, often ignore decency of cards from their decks and predict by the decency of card to card. A deck prediction can improve the existing approaches.

2.5.1 Evaluation of Hearthstone game states with neural networks and sparse auto encoding

The article of Jan Jakubik [34] describes an approach of evaluating game states in the game Hearthstone. Goal was the evaluation of game states by machine learning algorithm by usage of 2.5 million game state out of training samples of Monte Carlo search tree driven agents. Sparse auto encoding was used to encode the information of the Minions in play on the players side and the opponents side. These two vectors and vectors which describe the turn number, the player's stats, the opponent's stats, and the player's hand were representing the game state. A five layer deep neural network was trained with the training data to evaluate the game state and predict the game's result which was the output of a single output neuron.

The agent achieved the 5th place of the AAIA'17 data mining challenge out of 188 submissions. Main weakness was the insufficient representation of complex cards, i.e. cards with unique effects.

2.5.2 Hearthstone: Finding the optimal play

The thesis of Tom Drijvers [35] describes an approach of evaluating game states and guessing the opponents hand in the game Hearthstone. The aim was the development of an decision supporting system for the player.

The goal of the evaluation was the scoring of the game state like a human player would do. It was devided into five parts. Its parts were the *health score* for the players' health and armor, the **secret score** for secret spells on the board, the **hand score** for the number of cards in the players' hands, the **deck score** for the number of cards in the players' decks, and the **board score** for the number and kind of Minions on the board. They were combined by a weighted sum which was weighted based on the players' strategies by the player during the game or by the algorithm based on previously defined weights.

The hand prediction was realized by a fuzzy set which represented the possibility of cards in the opponent's hand. During start of the game each hard had the same degree of membership including cards which were not actually in the deck if the deck was unknown. The fuzzy set was updated after the mulligan face which is motivated by the assumption that a player will keep specific cards in their hand, e.g. cheap cards or cards with high impact on the game state during the first turns of the game. The next updates were done during the drawing of cards.

It was simulated in two steps. First, the playing of cards was tested. Second, possible attackers were simulated. Both step were evaluated by the their change of the game state. The CI was tested for player support and in fully automated mode. Opponents were the CI itself and other CI. The weighting and hand predictions were adapted during the tests. In the results, the CI preformed better than related algorithms in the play and the evaluation had an advantage over standard evaluation after the adaption.

The CI was not tested against actual players, like the hand prediction was build upon. The prediction of the opponent's deck was not considered, though, the opponent's deck was adapted after knowing it. The simulations was not made over multiple turns.

2.5.3 I am a legend: hacking Hearthstone using statistical learning methods

The CI of *E. Bursztein* [36] which uses Hearthstone card sequences to predict future cards was called 'game breaking' of the Hearthstone developer Blizzard. Its prediction is based on learning from game-replays which cards are played significantly more often then other cards and which cards are often played together. The data were extracted from debug logs [37] and used to generate replays. Bigrams were constructed on the sequence of played cards from the replays containing all combinations of cards, and used to construct a co-occurrence table conting all this co-occurrences happen. An ranked list of cards which are in the opponents deck and likely to be played was created for the prediction, while summing the frequency of card appearances of all retrieved cards that were co-played with the seen cards. *Burzstein* evaluated the algorithm by generating the ranked list and comparing them with the effectively played cards of future turns. Card prediction were mostly correct up to 95% in the turns three to five. The dependence of a card occurence to a complete deck was not considered.

Chapter 3

Implementation

CCGs are games, where information are hidden to an extent, though rules of the game and possible moves of a player are well known. Winning in a CCG requires the understanding of games and their theory, in general, to determine which type of game they are and how to classify them into game theory, so it is possible to develop a successful strategy. These topics are covered in section 3.1.

The selection of a test environment has an influence on the agent who is playing the CCG. So a CCG game has to be chosen before its development. Factors for the selection of a CCG, test environment candidates, and the final choice are covered in section 3.2.

After the selection of a CCG, an algorithm, which is part of the game playing agent, has to be build which tries to predict the opponent's strategy in a whole and detect risks of moves which are part of the strategy. These risks are the cards of the deck which is the opponent's strategy. Furthermore, if maximizing the number of total won is the goal of the agent, the will have to predict changes of winning a game, so the agent can decide whether to continue the game, if win chances are high or to end it if a loss is likely. Since stochastic conditions determine the cards of the opponent and its strategy, in combination with the metagame, they and the resulting problem are covered in section 3.3. Clustering of the metagame data which is the solution to this problem is the topic of section 3.4. The impacts on the stochastic conditions and the search on the Hierarchical Cluster Tree which was developed in the process are continued in section 3.5.

3.1 Playing Collectible Card Games

Collectible Card Games are a subgroup of card games with special characteristics that distinguishes them from other card games, described in Subsection 1.1.3. Their classification by game theory and the influence of their characteristics onto the strategies of playing the game are dealt with in the this section.

3.1.1 Classification by game theory

Game Theory is a scientific field which deals with the strategies or player actions and the rules in a game. The basics of game theory are discussed in 2.1. CCGs can be classified as follows.

CCGs are played in different play modes. Starting from 1-vs-1 games in all CCGs, modes such as the 2-vs-2 in games like 'Magic the Gathering' ('MTG') and 'Yu-Gi-Oh' are possible. Even advanced modes like deathmatches, where multiple players try to win against all other players, team matches and their combination are played in 'MtG'. Based on the mode, disparate types of cooperation exists in this game. Where 1-vs-1 are non-cooperative, other game modes offer a combination of being non-cooperative and cooperative, either enforced by the game rules or the result of player politics. Therefore, the 2-vs-2 mode is cooperative in the team of two and non-cooperative in the interaction with the opponent.

In a game with multiple players, the strategy in a CCG is defined by the deck, each player chooses, and the cards, it contains. The outcome of the game defines the payoff, typically either win or loss. In contrast to games like Rock-paper-scissors, the outcome of two players choosing their strategy is not specified by the game rules, which rate the strategies against each other, and thereby not predetermined. Instead, the payoff is a stochastic value, which is calculated by all outcomes of plays of all pairings of strategies, i.e., the number of wins in proportion to the number of losses, and hints the advantages of one strategy against another. This stochastic value is called the win rate of a strategy. It is also part of the metagame where it is calculated over all players and strategies. The game is symmetric since both players have the same choice between decks and the same chances of winning a game.

The game consists of multiple game steps, which have an own tactic and thereby an own payoff inside the game. The tactic is determined by a single move each player takes which changes the game state. A move requires resources, may it be in the form of real existing components like cards in all CCGs or in the form of virtual components like 'Mana' in games such as 'MtG' and 'Hearthstone'. The payoff of it can be calculated by the number of resources which are spent to make the move and the impact on the game state. Thereby, a card which destroys all creatures of the opponent has a higher payoff if the opponent has more creatures instead of a few. The payoff matrix is influenced by its input values. If the input is limited to the move, the payoff is not symmetric. If the input also contains the current game state, the payoff is symmetric. The same is applicable to the zero-sum characteristic of a game.

The Move-Order in CCG is sequential in the first place. The gameplay is divided in turns. Each player has an own turn where it is able to make moves which it can not do in another player's turn. Such as drawing a card, playing creatures or attacking another player or his creatures. Players are not allowed to make this move as often as they want though. The action is limited to a phase inside of a turn and the phase changes automatically after taking that action. This way, a player attacks only once each turn with its creatures. Besides taking a move, an opponent can take a move in reaction before the first move is processed; this is possible in games like 'MtG' or 'Yu-Gi-Oh'. These CCGs work with a move stack, on which a moved is placed before it is processed. A move in reaction is placed on top of previous moves and may influence it. The ability to react is regulated by switching the priority of each player after making a move. When the priority is switched and what moves a player can make is determined by the game rules. Triggered moves are moves which are made automatically after another move happened. For instance, a creature can have an ability which triggers these automated moves, like doing something if another card is played. Triggered moves may be processed simultaneously.

As it was mentioned in subsection 1.1, CCGs are imperfect information games. The information shared of all players is limited to the game state, including creatures and other objects on the game table, the resources and the number of cards of both players in the decks, the number of cards in the hands and the cards in the piles of played cards. Limited information to one player is the cards in its hand and a set of cards in its deck, not necessarily the order of cards in the deck. Therefore, hidden information is the cards of the opponent, in their hand and deck. Additional information is gained by the progress of the game and through special moves, which enable a player to look into his deck, or the opponent's hand or deck. In regards to the metagame, the information of which deck the opponent chooses is hidden.

Winning conditions in CCGs enforce the end of a game and give the win to the player who meets the condition. A Traditional winning condition is the reduction of the opponent's life points of the opponent in games such as 'MTG', Hearthstone' and 'Yu-Gi-Oh', the destruction of a certain amount of its creatures like in 'Pokemon TCG' and its drawing of all its cards in the deck. Card effects may introduce additional winning conditions. Since the number of cards in a deck is limited to a certain number, the game length is limited to that number or to an additional condition which reduces the number of life points each time a card could not be drawn like in 'Hearthstone'. The number of cards in a deck may change by card effects which, in consequence, alters the game length.

Since earlier discussed, the metagame collects the outcome of all strategies against each other and the times of strategies picked. Some strategies have a significant advantage against others. The outcome of a game is influenced by the metagame, since choosing a strategy, or deck, which is played less often and playing against a deck which is played more frequently, result in more wins if the win rate of the less played strategy is higher.

The process of creating a strategy which has advantages versus other strategies in CCG is explained in the following section.

3.1.2 Building a strategy

The higher the win rate of a strategy the more successful it is. A win rate of over 50% is aimed when building a new deck, i.e. it wins more games than it looses. Relevant factors and the process in building a strategy, or deck, in CCG is covered in this section.

To build a legal deck and to pick cards, a player has to adhere to the game rules of the game format. A game format has distinct rules, such as the total number of

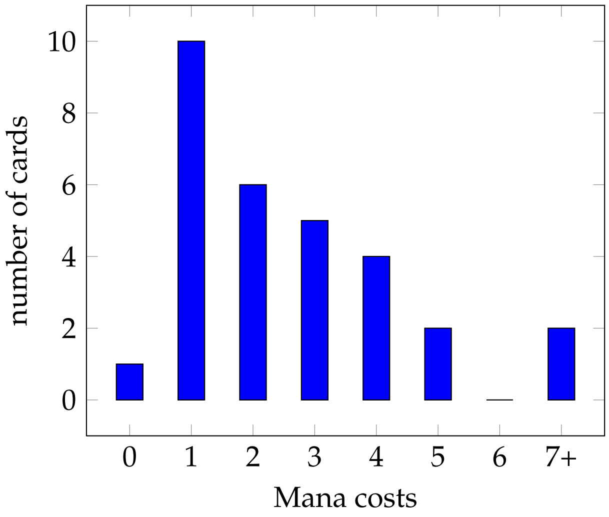

FIGURE 3.1: Mana curve of a Hearthstone deck

cards in a deck and the allowed number of specific cards. Furthermore, it defines the the player's available options during the game and the use of resources. After choosing the format, a player has to consider that its deck is playable. For instance, in Hearthstone, each card has its own Mana costs, the amount of Mana required to play that card. One Mana source is added to the player each turn, i.e. it can use one more Mana. If the player builds a deck which contains mainly cards of high Mana costs, the player is not able to play cards until he reaches the turn with the required Mana. Nevertheless, cards are always playable after waiting a certain amount of turns in Hearthstone, since the resources are produced automatically and not bound to a card. In contrast, games like MtG and Pokemon require resource cards which, after playing, are able to generate the resource. So a player needs the required number of resource cards to play resource consuming cards in these games. The ratio of cards indicates the ratio of resource cards and other cards. The optimal ratio is influenced by the Mana curve which also exists in CCGs which does not require resource cards. It is represented by a graph which describes the relation of card costs and number of cards, such as exemplary shown in figure 3.1.

A deck with a curve which concentrates on low Mana costs requires a small amount of resource cards. On the other way around, a curve which concentrates on high Mana costs requires a greater amount of Mana resources. A second characteristic that the Mana curve influences is the speed of a deck, the speed until it develops it full potential. Fast decks, with a small Mana curve, i.e. many cards with a low Mana costs, try to beat the opponent before it can start to take counter measures. Slow decks, with a higher Mana curve, i.e. cards with high Mana costs, try to service until they are able to play Mana intensive cards which can influence the game state heavily to their favour.

These strategies result in three main deck types which are successful in the metagame.

- The **aggro deck**, which is a fast deck and which only interaction with other strategies is trying to prevent moves of the opponent that prevent the own strategy.

- The **combo deck** builds around strong synergies of cards. It profits from the few interaction of aggro decks, since the winning synergy can be deployed earlier then the aggro strategy is able to beat the combo strategy player.
- The **control deck**, which is a slow deck and able to prevent the combo strategy from starting its synergies by control effects like removal of already played cards and prevention of playing cards.

It is likely to dominate the combo deck, however, likely to loose against a aggro strategy since it is not able to start sufficient counter measures before it is beaten.

The prediction of the opponents' moves during the game is a factor which determines when cards have to be played and Mana has to be to spend. It is based on the game state and dependent on the unknown cards of the opponent. A player can try to predict which cards can be played by the opponent by knowing all the possible cards which have the remaining amount of the opponents' Mana or less. On this knowledge, it build its following moves. Though, the prediction is impeded by the number of the opponents' remaining cards in hand. So keeping a healthy amount of cards in hand increases the chances of having a counter for the next opponent's move and the ability to bluff, like acting to have counter measures even if there are non, that raises caution of the opponent and, in the following, it may not play its cards with high game state impact. Playing cards with card draw effects and sometimes not playing cards, sustain a healthy amount of hand cards.

3.2 Selection of a test environment

In this section, different card games which were possible test candidates are presented and compared. Reasons for choosing Hearthstone, the structure of which is presented in Subsection 1.1.4, are explained.

Making a statement about the influence of the metagame of CCGs and their reciprocally influence on strategies, requires a CCG with a representative number of players and a sufficient number of test data. Though the exact number is unknown, Magic the Gathering (MTG) is commonly known as the biggest printed CCG world wide by player base. Nevertheless, it is an unfitting test environment, since strategies are irregularly tracked, most likely when a tournament takes place. Though, a metagame exists and strategies are more successful, the few amount of games played with this strategies results in unrepresentative data. More crucial than the test data is the missing software environment for using agents, which can play games against each other. This fact is the case for all printed CCG like 'Pokémon TCG' and 'Yu-Gi-Oh!', wherefore they are not looked at any further.

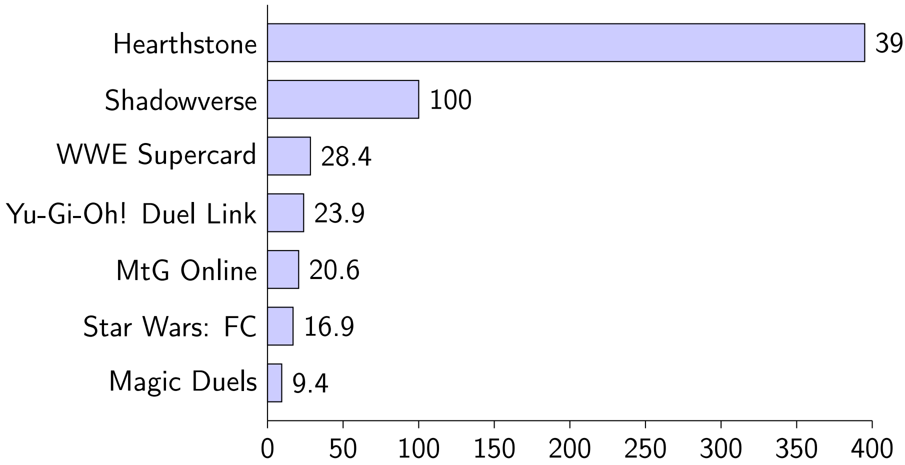

FIGURE 3.2: Leading digital collectible card game (CCG) titles worldwide in 2016, by revenue (in million U.S. dollars) [38]

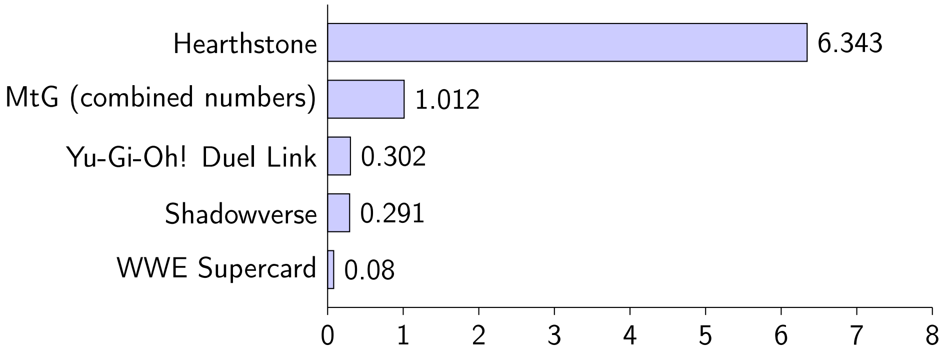

FIGURE 3.3: Leading digital collectible card game (CCG) titles worldwide in 2019, by follower base on streaming platform 'twitch' (in million people) [39]

According to statista, in the year 2016, the leading online digital card games by revenue were 'Hearthstone', followed by 'Shadowverse', the combined numbers of 'MtG' games, 'WWE Supercard' and 'Yu-Gi-Oh!', shown in 3.2.

The numbers are supported by the interest in these games which can be deduced by the follower numbers on the streaming platform 'twitch'. Except, 'MtG' which interest has grown in, since a new game released. The numbers of followers are visualized in 3.3.

In terms of this data, games with a representative number of players are 'Hearthstone' and 'MtG Arena'. Though, 'Hearthstone' has been released in 2014, it has a four years longer live time on the market than 'MtG Arena' and, therefore, the fan base was able to develop programs which use APIs to interact with the game, game tracking platforms and a test environment for 'Hearthstone' playing agents. The platform 'HSReplay' [18] and the program 'Sabberstone' [40] are described in the following.

- **HSReplay** is online platform which tracks data about Hearthstone games. When a player registers, it enters an API key in this platform to unable the

Property	**Meaning**
hero	One out of nine heroes
deck_id	Id of deck in database
deck_list	List of card-ids of cards in deck
archetype_id	Pre-assignment to strategy type
total_games	Total number of games the deck was played
win_rate	Rate of winning games
avg_game_length_seconds	Average length of a game in seconds
avg_num_player_turns	Average number of turns of the player

TABLE 3.1: Properties of a deck

Property	**Meaning**
card_id	Id of card in database
artist	Name of card's artwork's artist
cardClass	Affiliation to one out of nine heroes
collectible	Whether card can be collected by players or not
cost	Mana costs to play card
flavor	Back story of card
name	Name of card
playRequirements	Additional requirements to play the card
set	Set in which the card war released
type	Type of the card, e.g. minion, spell
attack	Attack value if a minion
health	Health value if a minion

TABLE 3.2: Properties of a card

platform to see and track all its games. For all 9 heroes, which 'hearthstone' offers, data are tracked which contain decks with all their cards, the times a deck is played, and the rate a deck wins. Also tracked are the overall win rate, game length, i.e. the average game time in seconds, and the average number of turns. All properties of a deck are visualized in table 3.1. 'HSReplay' also has a database for the cards, which contains its Mana costs, the hero who can use the card, artist, rule and flavour text. Furthermore, it contains the card health, its attack, type and rarity. The properties of a deck are visualized in table 3.2.

- **Sabberstone** is a 'Hearthstone' simulator, which enables game cloning, i.e. taking over of game states from the actual game to the simulator, and game tree search of 'Hearthstone' games. It is actively developed and new card mechanics are implemented after they released in the actual game. It offer a way for CI developers to test their agent in a simulator. Over 80% of all cards are implemented according to sabberstone (10.01.2019) [40].

Analyzing the metagame to predict possible cards of the opponent, his deck, and the chances of winning the game, requires either the tracking of data by an agent itself or by using tracked data of real users.

Tracking the data by an agent has some disadvantages. The agent has to have a starting point to work on, i.e. if it can not play the game, it can not track actual data. Moreover, it needs acceptance of the player base, i.e. it must be 'fun' to play against, so that the agent is able to play many games against many players to acquire a representative amount of data.

The starting point of a game playing agent can be, to let it play against itself. So it learns to play the cards, use their mechanics, and to build decks. That creates a metagame of CI versus CI. Though, this is a virtual metagame which is different from the real one, player versus player metagame, or at least not synchronized. The developed agent is able to play against players and to track data, where the following has to be taken into account, the players must not know that they are playing against an agent. Otherwise, the creation of a new metagame, players versus CI, is conceivable which also results in wrong card and deck predictions.

Since the agent should be able to play against real players, it has to use data on player-vs-player games. This makes the predictions realistic against real players. The benefit of using player data to predict upcoming cards and the opponents deck is that a game playing agent has not to be developed. Therefore, the tracked data of 'HSReplay' are used which offers a weight database of the metagame, consisting of the deck playing data and match-up results.

It is assumed that the data gained from 'HSReplay' are correct.

3.3 Stochastic conditions and problems

Playing a CCG is influenced by various probabilities which may be founded in statistical data. The stochastic conditions and problems which arise when predicting the opponent's strategy, are discussed in this section.

A CCG offers a collection, or set, of cards $K = \{k_1, k_2, \ldots, k_{n_k}\}$. A player can choose cards out of this collection to build a deck d which is a multiset of K as denoted in Subsection 2.4. A multiset of K is defined as $\mathcal{K}_h = \{\gamma_{\mathcal{K}_h}(k_i)/k_i : k_i \in K\}$ where $\gamma_{\mathcal{K}_h}(k_i)$ is the number of appearences, or count, of the card k_i in the multiset $\mathcal{K}_h$. A deck is a special multiset of K with a specific size, it is defined as $d_j = \{\gamma_{d_j}(k_i)/k_i : k_i \in K\}$. $D = \{d_1, d_2, \ldots, d_{n_D}\}$ is the set of all decks that are played. The allowed number of $\gamma_{d_j}(k_i)$ differs in various CCGs, though, in 'Hearthstone' it is restricted to $\gamma_{d_j}(k_i) \leq 2$. The size of a deck is the sum of cards in a deck.

$$s(d_j) = \sum_{k_i \in d_j} \gamma_{d_j}(k_i) \tag{3.1}$$

The deck size varies in different CCGs, though in this case it is exactly $s(d) = 30$. With the count of a card in a deck and the size of the deck, it is possible to calculate the probability of drawing this card.

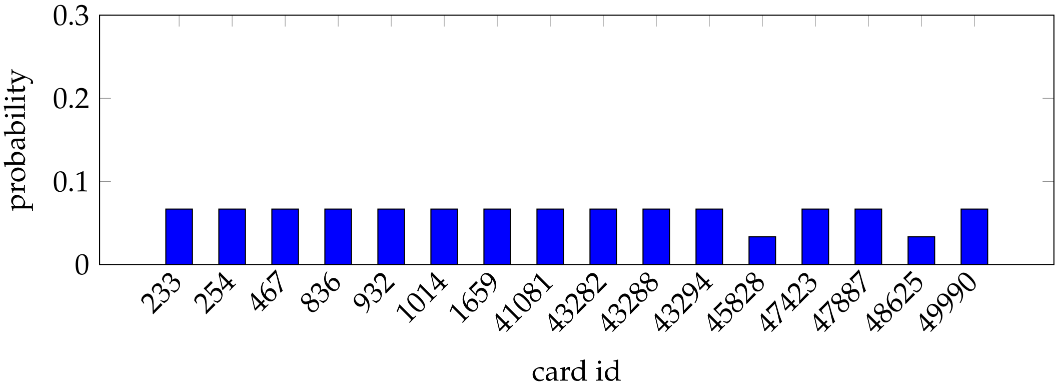

FIGURE 3.4: distribution of cards in a Hearthstone deck of the hero Druid

$$Pr(k_i|d_j) = \frac{\gamma_{d_j}(k_i)}{s(d_j)} \tag{3.2}$$

The probability of drawing the first card in a perfectly mixed deck is, e.g., distributed as shown in the figure 3.4.

Since it is a combination without repetition, the cards are removed from the deck after they are drawn, the probability of drawing the following cards is:

$$Pr(k_i|d_j \ominus \mathcal{K}_h) = \frac{\gamma_{d_j}(k_i) - \gamma_{\mathcal{K}_h}(k_i)}{s(d_j \ominus \mathcal{K}_h)} \tag{3.3}$$

With the information of the metagame which contain statistical data, the probability of a deck is calculated.

In the following the metagame is denoted as a multiset of games:

$$M = \{\gamma_M(d_j)/d_j : d_i \in D\}. \tag{3.4}$$

$\gamma_M(d_j)$ is the number of total games the deck is played in the metagame. Furthermore, the following can be calculated, the count of a card in the metagame considering a specific deck:

$$\gamma_M(k_i|d_j) = \gamma_{d_j}(k_i) \cdot \gamma_M(d_j) \tag{3.5}$$

and the count of a card in the metagame in general:

$$\gamma_M(k_i) = \sum_{d_j \in D} \gamma_M(k_i|d_j) \tag{3.6}$$

When playing against a known player, data from the previous games determine which deck it will pick to play. Equivalent, the total games of a deck in the metagame determine the probability of the deck when playing against a random opponent. The probability it is played is

$$Pr(d_j) = \frac{\gamma_M(d_j)}{\sum_{d_z \in D} \gamma_M(d_z)} \tag{3.7}$$

Furthermore, the probability determines the overall probability that a specific card is drawn. Since all decks have the same size, it is possible to change the equation into another form.

$$Pr(k_i) = \sum_{d_j \in D} Pr(d_j) \cdot Pr(k_i|d_j) = \frac{\gamma_M(k_i)}{\sum_{k_z \in K} \gamma_M(k_z)} \tag{3.8}$$

Seeing a card, enables determining the a posteriori probability of the decks. So decks which contain the card more often, are much more likely, than decks with fewer occurrences of this card. The probability of decks which does not contain the card becomes zero.

$$Pr(d_j|k_i) = \frac{Pr(d_j) \cdot Pr(k_i|d_j)}{\sum_{d_z \in D} Pr(d_z) \cdot Pr(k_i|d_z)} = \frac{\gamma_M(d_j) \cdot Pr(k_i|d_j)}{\sum_{d_z \in D} \gamma_M(d_z) \cdot Pr(k_i|d_z)} \tag{3.9}$$

The probability of the next card, is also calculated by the already seen cards. The probability of it in a deck, has to be adjusted by removing the seen cards K_h from the decks, resulting in $\gamma_{d_j \ominus \mathcal{K}_h}(k_i)$ as shown in Equation 2.19.

$$Pr(k_i|k_h) = \sum_{d_j \in D} Pr(d_j|k_h) \cdot Pr(k_i|d_j \ominus \{k_h\}) \tag{3.10}$$

Some problems arise when following this procedure to calculate card and deck probabilities and predict cards based on this method which is called the stochastic method from here onwards. If a card of a set of the played card is not contained in a deck, the probability of this deck will become zero, even if all other cards are contained in the deck. This has also an influence on the card probabilities. So decks which are not registered in the metadata, i.e., a combination of cards is played which does not exist in any deck in the metadata, can impede the probability calculation of decks and cards. Furthermore, some effects of cards cause the possible existence of deck external cards inside the deck or hand of the player, or the existence of a third deck internal card inside the deck, which also changes the deck size and impedes the calculation. Though the problem of seeing unknown cards could be solved through a Laplace smoothing by adding a small pseudo count of each possible deck [41], this would increase the computational costs. The number of added decks for each of the nine Heroes would have a lower bound of $\binom{400}{30} > 1.4 \cdot 10^{45}$ on the assumption that a card could be contained once. The real number is bigger since some cards are allowed twice in a deck.

A solution is required which deals with unknown cards, keeps the computing costs to such a small extent that it can be used inside the game. Distinguishing between a new deck and existing decks, which contain added cards by card effects, is a task which has to be solved.

3.4 Clustering of the meta data

Problems of the previously shown procedure in section 3.3 are the fix deck size and the exclusion of decks, after seeing unknown cards, because of the stochastic calculation of decks and card probabilities. Solving this problem is focused in this section. Procedures to cluster decks and cards are presented.

The clustering algorithms which are discussed in the section 2.2.2 are used to categorize the decks into groups which contain a higher card variety, making unknown cards less likely. Since decks may be played which are not in the clustered data and differ in cards of two decks in the same cluster, though, all cards of the deck are in the cluster, this way, these decks and their cards are still calculable. So the unknown strategy can be approximated by knowing similar strategies. The calculation of the probabilities in the cluster structure is topic of the next section 3.5.

Clustering can be achieved on different ways. The following clustering algorithms are considered:

- **k-means** which is described in Subsubsection 2.2.2.1 is a simple approach, though, offers good solutions for many clustering tasks. However, characteristics k-means has cause problems which occur when trying to cluster decks. Strategies can be contracted in many ways, the number of strategy groups is uncertain, so it is not possible to give the algorithm the number of clusters, it has to work with. However, this can be solved by starting multiple runs with a different number of initial clusters. It is possible, that strategies appear which are played rarely and consist of completely different sets of cards than other strategies. These are outliers which k-means is not able to handle. In contrast to strategies which exist between two strategy type extremes, it is resulting in non-spherical clusters. Thinking of two decks with different cards which have quite similar effects, cards can be exchanged between the decks without harming the strategy. Considering these factors, k-means is not a suitable candidate for the deck clustering task.

- **DBSCAN** which is described in subsection 2.2.2.3 solves the problem of non-spherical clusters and of the number of clusters, since it does not require the number before starting the clustering process. Decks can be scattered in areas of high and low density, resolving in problems with clustering data of different densities. The selection of its parameters is another disadvantage since it has to be selected carefully. There are extensions for DBSCAN which solves these problems, such as hdscan [42] and aodbscan [43]. However, with this expansions, it is still not suited as clustering algorithms for the deck clustering task, since its density based procedure, is not representative of the data.

- **Hierarchical Clustering** whose basics are summarized in subsubsection 2.2.2.2 has all requirements to solve the clustering task. In contrast to k-means, it is able to handle outliers, an unknown number of clusters and non-spherical clusters under specific requirements, such as edgecut, dendrogram analysis, and their linkage method. In addition to DBSCAN, it clusters data of various densities. Furthermore, its clustering process offers a way to organize the strategies that are useful in the later search of cards, decks, and their probability calculation. Since the agglomerative, hierarchical clustering can create a binary tree, which contains clusters of increasing hardness since the root cluster contains cards of all decks, it is soft, and leaf clusters contain cards of only one deck, it is hard. The created tree is called the hierarchical cluster tree (HCT) onward.

Hierarchical clustering is the algorithm of the previously listed ones which fits the task, of clustering the decks and their cards, best. Since in the game 'Hearthstone' each player has to pick a hero and build a deck around it and each player knows the opponent hero, the decks which are played in the metagame are clustered by the hero in the initial step. Since nine heroes exist, it results in nine following clustering procedures. The clustering steps which are done, are described in following.

1. Since the clustering is agglomerative, a cluster is created for each deck, the decks are the data points. The data points are a vector $v = (v_1, v_2, \dots, v_{n_K})$ which number of dimensions are equal to the number of permitted cards n_K for the specific hero. So, for a deck $d_1 = \{2/k_1, 2/k_2, 1/k_3, \dots\}$ the vector is of the form $v_1 = (2, 1, 1, 0, \dots)$ and for a deck $d_2 = \{1/k_1, 1/k_2, 1/k_3, 1/k_4, \dots\}$ the vector is of the form $v_2 = (1, 1, 1, 1, 0, \dots)$. The information about the deck's total games and its cards can be accessed.

2. The distance for each pair of clusters is calculated based on the linkage method, resulting in a distance matrix. This means for

   - single linkage, that for each cluster pair the distance between every deck of this pair is measured, either Euclidean or Jaccard distance is used. The shortest distance between the data points is the distance of the clusters. If v_1 is in cluster c_1 and v_2 is in cluster c_2 and their distance is the smallest distance among the decks of both clusters, the distance $dis(c_1, c_2) = dis(v_1, v_2)$ is the cluster distance.
   - complete linkage, that the distance between every decks of two clusters is measured as in single linkage. The largest distance between data points is the distance of the clusters.
   - average linkage, the pairwise distance of each deck of two clusters is measured as in single linkage. The average is calculated which is the distance of the two clusters.
   - centroids, a center for the data points is created. If d_1 and d_2 are in the same cluster the center of their vectors v_1 and v_2 is $v_{c_1} = (1.5, 1, 1, 0.5, 0, \dots)$. The cluster distance between v_{c_1} and an other cluster center v_{c_2} is $dis(c_1, c_2) = dis(v_{c_1}, v_{c_2})$.

- ward, the centers for the data points of clusters is created. The deviation of the distance of the points of two clusters to their cluster center and the distance of the points to the simulated common cluster center of the two clusters is measured. For instance, the distance deviation of for a cluster $c_1 = d_1$ and a cluster $c_2 = d_2$ is $dev(c_1, c_2) = \sqrt{2}$, since the distance of the data points to their center is 0 and the sum of their distance to their simulated common center $v_{c_{1,2}}$ is $dis = \sqrt{0.5} + \sqrt{0.5} = \sqrt{2}$. This is proceeded for each cluster pair, resulting in a matrix of deviations. Small deviations hint close clusters.

3. A new cluster is created for the closest two clusters. Both clusters are assigned as children to the new cluster and their decks are linked to it.

4. Continue with step 2 until there is only one cluster remaining.

3.5 Search in a hierarchical cluster tree

After the construction of the HCT, the aspect to consider next, is the search on the tree. In this section, the problem of unknown decks and cards is solved, and probabilities of cards which might be played next are calculated based on the most likely cluster in the tree. Furthermore, the questions of the opponent's most likely deck/strategy and the chance of winning the game are answered. The solution approach is a search on the HCT to find the most likely cluster of decks. The search always starts from the root of the HCT.

The HCT is a tree of clusters. C is the set of all cluster, whereby $C = \{c_1, c_2, \ldots, c_{n_C}\}$. According to the done steps in the previous section, a cluster is created from a deck or is a merge of two clusters, it can be record that:

$$c_q = c_r \bigcup c_o \vee \{d_q\} \tag{3.11}$$

Three steps are added after the construction of the HCT to enable probability calculation free from the decks and find the most likely cluster. They are describes further down below.

1. Restructuring the data

2. Traversing the HCT until determination criteria is met

3. Calculate card and deck probabilities

The ranked list which is the result of the calculation of all card probabilities is adjusted for the card prediction by the available Mana of a turn and removing of the expensive cards. The adjusted ranked list contains the predicted card for a turn sorted by probability. The prediction of cards with the HCT is called HCT method from here onwards.

3.5.1 Restructuring the data

The calculation of cards, independent of the deck, required a new data structure. So clusters get a weight based on the sum off all total games of the decks, they contain, called size s of a cluster.

$$s(c_q) = \sum_{d_j \in c_q} \gamma_M(d_j) \tag{3.12}$$

Also, a cluster has information about the cards in the decks. To maintain ratio between cards over each deck, the count cards in a cluster is used instead of its average number in a deck, ignoring the total games. The count of a card in a cluster can be defined as the sum of its count in the metagame considering specific decks contained the cluster.

$$\gamma_{c_q}(k_i) = \sum_{d_j \in c_q} \gamma_M(k_i | d_j) \tag{3.13}$$

Since the probability of cards changes if a card has been played, the number of the card has to be maintained in the cluster. Therefore, the card information are saved in two sets, the first set Υ_{2,c_q} containing all cards which are in decks twice and the other set Υ_{1,c_q} containing all cards which are in decks once in the cluster c_q. Furthermore, the count of a card in a cluster which is contained once γ_{1,c_q} or twice γ_{2,c_q} in decks is known.

$$\gamma_{\iota,M}(k_i | d_j) = \begin{cases} 1 & \text{if } \gamma_{d_j}(k_i) = \iota \\ 0 & \text{else} \end{cases} \tag{3.14}$$

$$\gamma_{\iota,c_q}(k_i) = \sum_{d_j \in c_q} \gamma_{\iota,M}(k_i | d_j) \tag{3.15}$$

$$\Upsilon_{\iota,c_q} = \{\gamma_{\iota,c_q}(k_i) / k_i : k \in K\} \tag{3.16}$$

The probability change is covered further down below.

$$t(k_i | c_q) = t^1(k_i | c_q) + t^2(k_i | c_q) \tag{3.17}$$

3.5.2 Traversing the HCT

Since the search is started from the root, the probability of cards and decks are broadly diversified. This changes, when cards are played. One child cluster can become unlikely, because of the seen cards. If this is the case the tree will be traversed to a more likely child cluster. This way, decks are removed from the field of view, so the probability of cards and decks become less diversified. When the tree is traversed, the card and deck probability alters. The probability of a child cluster c_r is determined by its size and the size of the other child cluster c_o, the total games of the parent c_q are $c_q = c_r + c_o$.

$$Pr(c_r) = \frac{s(c_r)}{s(c_q)} = \frac{s(c_r)}{s(c_r) + s(c_o)} \tag{3.18}$$

If a card is played it will influence the probability of child clusters. For the calculation, the probability of a card in a cluster is required.

$$Pr(k_i|c_q) = \frac{\gamma_{c_q}(k_i)}{\sum_{k_z \in K} \gamma_{c_q}(k_z)} \tag{3.19}$$

The probability of child cluster behaves the same as the deck probability in the equation 3.9.

$$Pr(c_r|k_i) = \frac{Pr(c_r) \cdot Pr(k_i|c_r)}{Pr(c_r) \cdot Pr(k_i|c_r) + Pr(c_o) \cdot Pr(k_i|c_o)} = \frac{s(c_r) \cdot Pr(k_i|c_r)}{s(c_r) \cdot Pr(k_i|c_r) + s(c_o) \cdot Pr(k_i|c_o)} \tag{3.20}$$

If a card is played which is not in the cluster, it can be ignored, since it has not any influence on the probability of neither child clusters.

The algorithm in the HCT traverses to a child cluster c_r if it is more likely than its sibling cluster c_o, the cluster's probability must met threshold $T = (0, 1]$, $Pr(c_r) \geq T$. The threshold functions are evaluated in chapter 4. The traversing stops if the threshold is not reached.

3.5.3 Calculate card, deck and win probabilities

After finding the most likely cluster, the probability of a cards is calculated. This is also possible anytime before the traversing determined, with the cluster at time.

Since the total game of a cluster is the sum of the total games of is children or all the decks it contains. The probability of a deck in a cluster is:

$$Pr(d_j|c_q) = \frac{\gamma_M(d_j)}{s(c_q)} \tag{3.21}$$

As mentioned in section 3.3 probabilities are changed by the played cards. When a card is played, it occurrence in Y_{1,c_q} is removed and it occurrence in Y_{2,c_q} is moved to Y_{1,c_q} which changes the probability of the next card, the removal of cards in a cluster is donated as $c_q/\mathcal{K}_h$. Cards which are not the cluster can also be ignored here, since they have no influence to the probability of the other cards to each other.

$$Pr(k_i|c_q \ominus \mathcal{K}_h) = \frac{t(k_i|c_q \ominus \mathcal{K}_h)}{\sum_{k_z \in K} t(k_z|c_q \ominus \mathcal{K}_h)} \tag{3.22}$$

Analogously, the probability of a deck changes like in the equation 3.9.

$$Pr(d_j|c_q,k_i) = \frac{Pr(d_j|c_q) \cdot Pr(k_i|d_j)}{\sum_{d_z \in D} Pr(d_z|c_q) \cdot Pr(k_i|d_z)} = \frac{\gamma_M(d_j) \cdot Pr(k_i|d_j)}{\sum_{d_z \in D} \gamma_M(d_z) \cdot Pr(k_i|d_z)} \tag{3.23}$$

3.6 Predictions

Based on the archetype labels in the metadata, the following prediction can be made on the clustering.

- Distinct cluster exist for each hero.
- Cluster with multiple decks exist.

The following predictions can be made for the in Section 3.3 discussed stochastic card prediction method and the HTC method which is discussed in Section 3.5. Since cards cannot be played in turn where players have less Mana than the required Mana costs of the cards, these cards can be filtered from the prediction to get the most likely upcoming cards.

- Many cards are predicted correctly due to the high number of unplayable cards.
- The card prediction becomes less accurate due to the few number of played cards and the increasing number playable cards. More cards are playable, since each turn, one Mana is added which can be used for playing cards.
- Due to the high number of played cards, specific decks become unlikely and an increasing number of cards is predicted correctly.
- After multiple cards are played, the stochastic method's correct card predictions decrease, since the method cannot assign a deck to the cards if the combination of cards is not contained in a deck.

Chapter 4

Evaluation

The Evaluation process can be split into two parts. First, is the clustering and generation of the HCT. And second, is the deck and card prediction including the selection of test data and evaluation of the predictions in regards to accuracy. Both parts are described in the following Sections 4.1 and 4.2.

4.1 Clustering evaluation

The Evaluation of the clustering is subdivided into four sections. Subsection 4.1.1 is about the selection of the test data. In Subsections 4.1.2 cluster results are generated and validated with the measures V-measure, homogeneity and completeness in Subsections 4.1.3. The in the validation made observations are discussed in Subsection 4.1.4.

4.1.1 Criteria for the test data selection

As described in Section 3.2, the online platform *HSReplay* offers a great amount of collected data from players. The interesting data are the deck data since they contain information about the times they were played what is important for the metagame. The data were collected by HTTP requests to the HSReplay server database. The request for the deck data accepts four parameters, which were chosen as the following to contain as much data as possible from the current metagame:

- **GameType** was selected as 'Ranked Standard', since the format is most actively played and allows the metagame to go through its phases, in contrast to the 'Wild' format where the metagame stagnates in the meta solved phase. It is more likely that decks of the metagame are played in 'Ranked' than in the 'Standard Play Mode'.
- **RankRange** was selected as 'all', so the data contains games of all games of all players independent of their rank.
- **region** was selected as 'all', containing the game from players of all regions, which are America, Europe, and Asia.

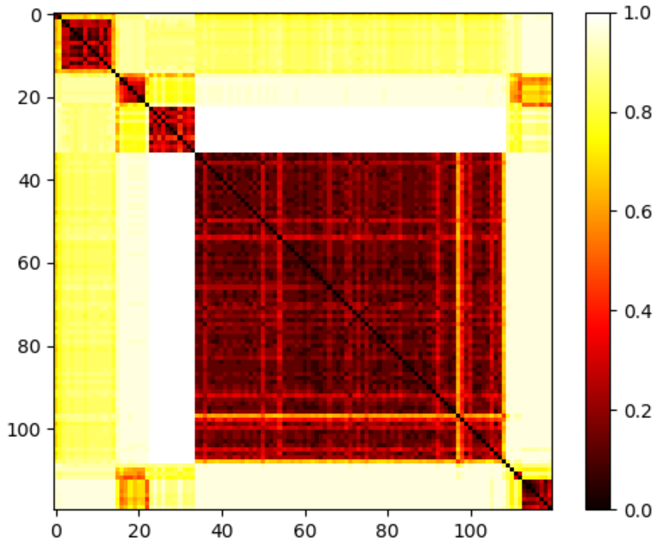

FIGURE 4.1: Distance matrix of 'Paladin' decks with Euclidean distance presorted by archetype labels

- **time range** was selected as 'current patch', containing only games of current rules set ranging from the 5th February 2019 to 20th February 2019. Since another rule set might have another metagame, it is not sensible to choose data ranging over multiple.

The data set only contains decks with at least 400 played games and up to a five-digit number. No further changes were done the data. Furthermore, the decks are labeled with their archetype by expert players.

4.1.2 Clustering the Data

The way to cluster the decks was described in Section 3.4. Building upon this procedure, clusters are build for each combination of the the distance functions Euclidean distance and Jaccard distance, and the five linkage methods single linkage, complete linkage, average linkage, centroids and Ward's method. This results in 10 combinations and, since the clustering is done for each of the nine heroes, in 90 HCTs.

According to the second step in the clustering procedure, the distance matrix is created. Figure 4.1 shows the results of a Jaccard distance calculation for decks of the hero 'Paladin' in heat map plots. Distances in the plot range from 0 to 1. Decks are presorted by their archetype labels. The distance matrices of all heroes is shown in the Figure A.1 and Figure A.2.

Clusters are already recognizable in this measure when decks are sorted by there distance to each other, which creates squares inside the plot. Some incongruities inside the squares are visible where the distance of a deck to other decks in the same square deviates.

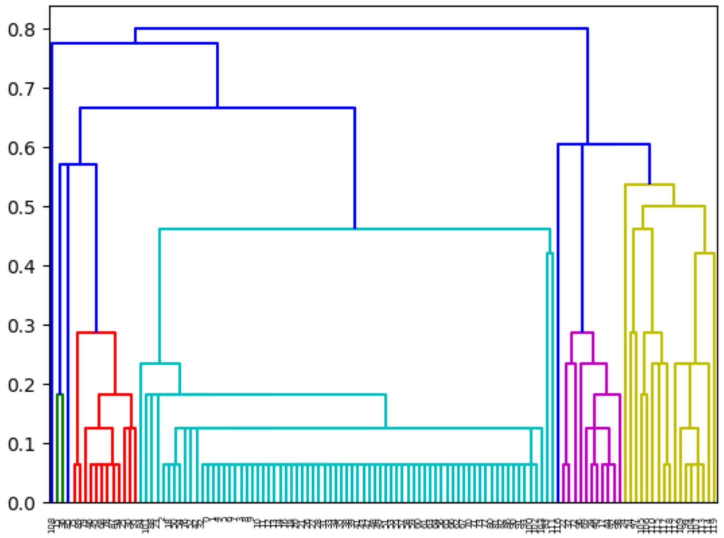

FIGURE 4.2: Dendrograms of the clustering of 'Paladin' decks with Jaccard distance and single linkage

Next up is the actual clustering of the decks. Figure 4.2 shows the dendrogram plot of the clustering with single linkage using the Jaccard distance on the decks of the 'Paladin'. Dendrogram plots of the all linkage methods are shown Figure B.1. Depending on the linkage method and distance function, the dendrogram looks differently. Single linkage tends to create chains when merging clusters, resulting in a vast amount of clusters which are merged in the dendrogram representation at the same distance. The other linkage methods have more branches in their dendrogram since the size of the merged clusters is more critical as in comparison to single linkage. Since Ward's method prefers to merge clusters with a small number of decks, the corresponding tree has the fewest levels of all methods.

4.1.3 Cluster validation

The previously noted, archetype labels of the decks enable a comparison of the archetypes and the clusters which are constructed on the decks in the data set. To do this, the external validation measures V-measure, homogeneity and completeness are used. It was described by *A. Rosenberg* on a conference[44].

- The **homogeneity** h is satisfied if clusters only contain data points from the same class, which is for example the archetype label. In worst case, each cluster contains every class. Considering a clusters $A = \{d_1, d_2, d_3\}, B = \{d_4, d_5\}$ and $C = \{d_6\}$, the homogeneity is met, if e.g. the corresponding labels are $l(d_1) = l(d_2) = l(d_3)$ and $l(d_4) = l(d_5) = l(d_6)$.

  How close a cluster is to the ideal class, is determined by examination of the conditional entropy of the class distribution given the proposed clustering.

This value is in a perfect homogeneous case $H(C|K) = 0$. Since this value is dependent on the data set size and distribution class size, it is normalized by maximal reduction entropy the data could provide $H(C)$, resulting a best-case result of $\frac{H(C|K)}{H(C)} = 0$ and a worst case of $\frac{H(C|K)}{H(C)} = 1$. In the degenerate case that $H(C) = 0$, the homogeneity is defined as 1.

$$h = \begin{cases} 1 & \text{if } H(C) = 0 \\ 1 - \frac{H(C|K)}{H(C)} & \text{else} \end{cases} \tag{4.1}$$

where

$$H(C|K) = -\sum_{k=1}^{|K|}\sum_{c=1}^{|C|} \frac{a_{ck}}{N} log \frac{a_{ck}}{\sum_{c=1}^{|C|} a_{ck}} \tag{4.2}$$

$$H(C) = -\sum_{c=1}^{|C|} \frac{\sum_{k=1}^{|K|} a_{ck}}{N} log \frac{\sum_{k=1}^{|K|} a_{ck}}{N} \tag{4.3}$$

- The **completeness** c is symmetrical to the homogeneity. It is satisfied if all data points with the same label are in the same cluster, such as the labels $l(d_1) = l(d_2) = l(d_3), l(d_4) = l(d_5)$ and $l(d_6)$ are in the clusters $A = \{d_1, d_2, d_3\}$, $B = \{d_4, d_5, d_6\}$. In worst case, each label is distributed among all classes.

  The distribution of classes to each cluster can be measured. The degree of their skew to each cluster is evaluated by calculating the conditional entropy of the proposed cluster distribution given the class of the component data points, $H(K|C)$. In best case $H(K|C) = 0$ and in worst case $H(C|K) = H(K)$. The normalization by $H(K)$ used to calculate c, considering the the degenerate case, H(K) = 0.

$$c = \begin{cases} 1 & \text{if } H(K) = 0 \\ 1 - \frac{H(K|C)}{H(K)} & \text{else} \end{cases} \tag{4.4}$$

where

$$H(C|K) = -\sum_{c=1}^{|C|}\sum_{k=1}^{|K|} \frac{a_{ck}}{N} log \frac{a_{ck}}{\sum_{k=1}^{|K|} a_{ck}} \tag{4.5}$$

$$H(C) = -\sum_{k=1}^{|K|} \frac{\sum_{c=1}^{|C|} a_{ck}}{N} log \frac{\sum_{c=1}^{|C|} a_{ck}}{N} \tag{4.6}$$

- The **V-measure** V is the harmonic mean of homogeneity and completeness. In the best case, when clusters arbitrary match classes, it is 1.

$$V_\beta = \frac{(1+\beta) \cdot h \cdot c}{(\beta \cdot h) + c} \tag{4.7}$$

The value β is the weight which weights homogeneity more strongly, when it is greater than 1, and weights the completeness more strongly, when it is less than 1.

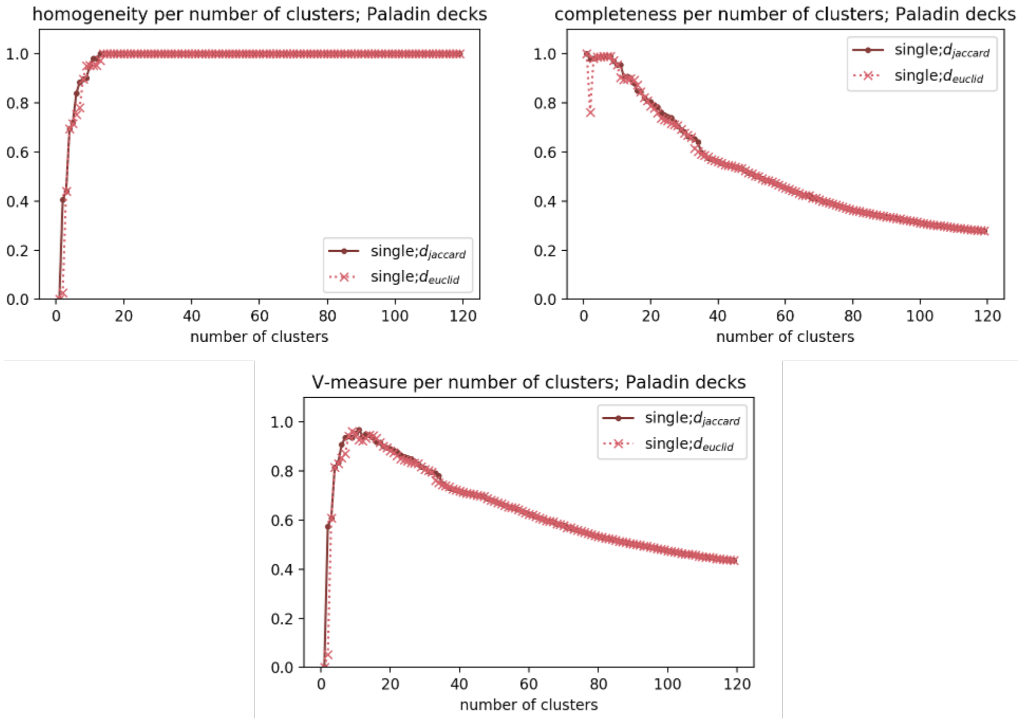

FIGURE 4.3: Homogeneity (top left), completeness (top right) and V-measure (bottom) of the clustered 'Paladin' decks and archetype labels

Since these measures are independent of the number of classes and clusters, and the used clustering algorithm. It can be applied to evaluate all clustering algorithms on a uniform scale.

For the validation of the clustering, labels of the clusters are extracted level-wise, since a common determination criterion is needed. The maximal distance cannot offer this criterion, since it is inconsistent over the distance function and linkage methods, e.g., the value of the Jaccard distance is between 0 and 1. It is validated for each of the in Subsection 4.1.2 chosen combinations and each extracted cluster level. Figure 4.3 shows the results of the clustering validation of 'Paladin' decks using single linkage. Compared are euclidean distance and Jaccard distance. The Figures C.1 and C.2 show additional V-measures.

In general, the Jaccard distance reaches faster a better homogeneity than the Euclidean distance. Furthermore, in general, it retains longer good completeness than the Euclidean distance, resulting in overall better V-measure. From the five linkage methods, the single linkage achieves the best results for the 'Paladin' decks, since its V-measure reaches a higher maximal value, i.e., the clusters match the archetype labels best, and keeps a better V-measure with an increasing number of clusters. The maximal V-measure is reached with eleven clusters, though, the number of archetype label is nine, what is possible, since the maximum homogeneity is reached with more than nine clusters. So, in comparison of all combinations, the combination of Jaccard distance and single linkage is the best-suited one for clustering of the 'Paladin' decks.

4.1.4 Discussion

From the in Subsection 4.1.2 created distance matrices can be deducted that the labels on the decks do not match the real clusters perfectly which is represented as deviations inside the squares. A wrong allocation might cause this. The properties of the created clusters are as predicted. The in Subsection 4.1.3 made validations show, that for the clustering of decks with the Jaccard distance generally scored better in regards to their V-measure in contrast to the Euclidean distance. In consequence, the Jaccard distance is more suited for the clustering. The clustering of the different heroes shows that different linkage methods score better in regards to their V-measure. For Example, Ward's minimum variance is most suited for the 'Druid' decks, and single linkage is most suited for 'Paladin' decks. Despite this, the V-measure is not a sufficient measure to show that the clustering was incorrect, especially, since some decks might have wrong labels. Furthermore, it is possible, that the prediction of the cards and decks is less depending on the best clustering and instead on the depth of the tree. That is why all linkage methods have to be considered for the following evaluation step.

4.2 Prediction analysis

In Section 3.3 and Section 3.5, methods to predect cards and decks where explained. In this Section, the validation of these methods is themed. Test data are selected in Subsection 4.2.1 and results of the card prediction with and without the HCT of Subsection 4.2.2 are validated in Subsection 4.2.3 and the observations of it are discussed in Subsection 4.2.4.

4.2.1 Prediction test data

For the validation of the prediction, replay data are in the focus of interest, since they are records of games and offer a sequence of played cards. *HSReplay* does not track the actual replay of the player games which are used to create the metagame data. So the data have to be collected from another source. Here, two sources are chosen.

- At first, the creation of replays out of the metagame data. Suppose, the metagame reached the meta solved phase, and mainly decks of the metagame are played, including mixed strategies which are combinations of decks out of this meta. The accuracy of the algorithms can be tested on this simplified scenario, to evaluate whether the clustering was meaningful for the prediction. One the one hand, this scenario differs from actual play scenarios. On the other hand, it offers the opportunity to generate replays including more turns and cards than actual existing.

  Considering the Mana costs of each card and the available Mana for each turn a sequence of cards can be created from the meta decks or from the decks which are created from a cluster of these meta decks. A card or a subsequence of cards in the sequence cost a maximum amount of Mana, corresponding to the turn. For the replays, sequences are generated over simulated turns. Since in the actual play sometimes no cards are played in a turn, this might influence the sequence. An algorithm creates 1.000 sequences of cards for each hero to test with.

1. It chooses a cluster at random, from a list of clusters including all clusters from the clustering process until a maximum distance was reached.
2. From this cluster, it adds a playable random card to the sequence for the turn. The chance that a card is added for a turn is 75%. A card is playable if its Mana is less than or equal to the remaining Mana for the turn and is was not played its maximum number yet.
3. The chance for each additional added card for the turn is 50%.
4. If no next card is added, the algorithm continues from step 2 for the next turn.
5. The sequence is complete if 15 turns were processed. This number is sufficient since replay data showed that the games usually did not last longer than this number.

The prediction of the deck can be compared to the actual deck or cluster it was created on.

- Second, using actual replay data. Software which enables a player to records its games is the 'Track-o-Bot' [45]. The replays are collected on a public repository called 'Collect-o-Bot' [46]. It contains anonymized replay data which are free to use. Since the data are not from HSReplay, the played decks might differ from the metadata of HSReplay and suggest a different metagame. So the weights of a deck (total games) might differ from the decks. Furthermore, the played decks in the replay data might not even be in the HCT. So cards can appear which are not in the on HSReplay trained HCT either. A replay consists of the play of 2 players. One card sequences can be generated out of it since one player gets the card 'Coin'. This behavior is a way of the game to compensate one player for the disadvantage of not starting the game. The card effect, adding temporal Mana, interferes with the card prediction. Nevertheless, the card can be handled by an agent which uses the algorithm, i.e., the agent can change the card prediction based on the card rules. To simplify the data further, sequences containing cards with one of the words 'hand', 'add', 'shuffle', 'into', 'deck', and 'discover' in their rules text, are removed since they are highly likely to change the card order or the cards in the deck and hand of the player. In a real game, an agent can adjust the algorithm based on the card, such as for the effect 'create a copy of a minion and shuffle it into your deck' the agent can replicate precisely this action.

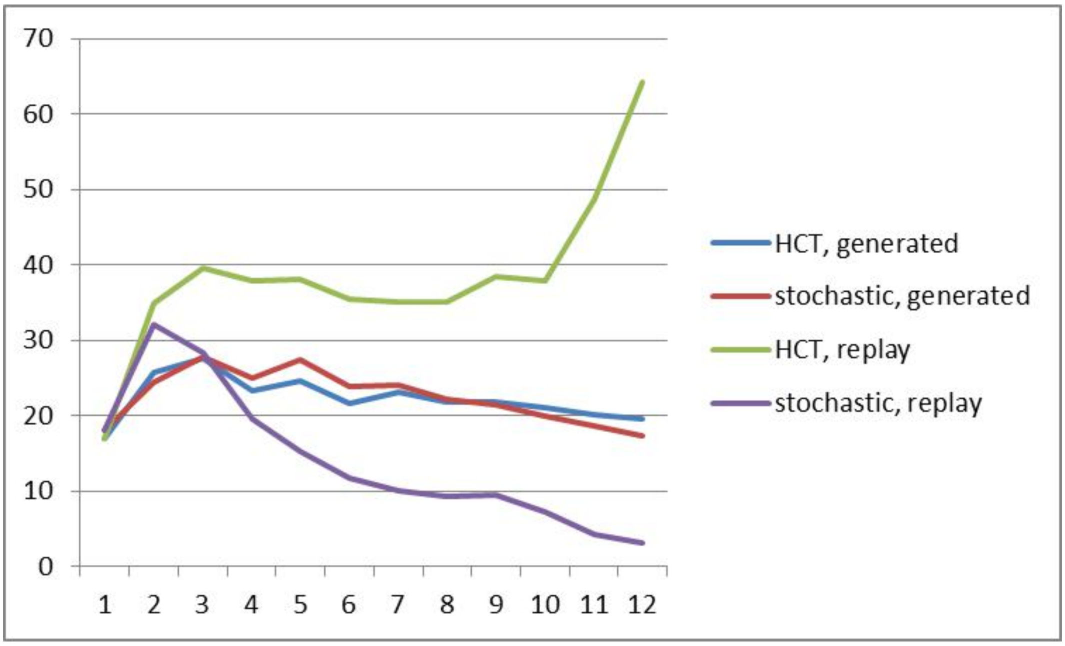

FIGURE 4.4: Average number of considered cards of 'Paladin' using the stochastic method and HCT methods with generated card sequences and sequences from replay data

4.2.2 Card Prediction

On both sets of sequences, the pure stochastic method, as discussed in Subsection 3.3, and the HCT method, as discussed in Subsection 3.5, are compared. For this purpose, the probability of each card is predicted turn-wise for each sequence, and the number of cards and decks is recorded. Input parameters are the cards of the previous turns.

The in Section 4.1 evaluated HCTs are used for the prediction with the HCT method. Jaccard distance was chosen as distance function for all of them in the following used HCTs. Single linkage is the chosen example of the following prediction. The threshold for the HCT traversing is chosen to be 0.99. Further evaluation of the threshold values takes place in the next Subsection 4.2.3. The following predictions are made on card sequences of 'Paladin' decks.

Figure 4.4 shows the average number of considered cards per turn using the stochastic method and the HCT method with generated sequences of cards and replay sequences.

The number of considered cards is increasing up to turn three for both card prediction methods and card sequences. Both methods predict approximately the same number of cards per turn for the generated card sequence. The number of predicted cards is slightly decreasing with progressing turns.

Significant divergences appear in the course of the curves of the replay card sequences. While the number of considered cards in the HCT method is continuously more significant in comparison to the number when using generated data and keeps a constant value after turn three, the number of considered cards is actively dropping

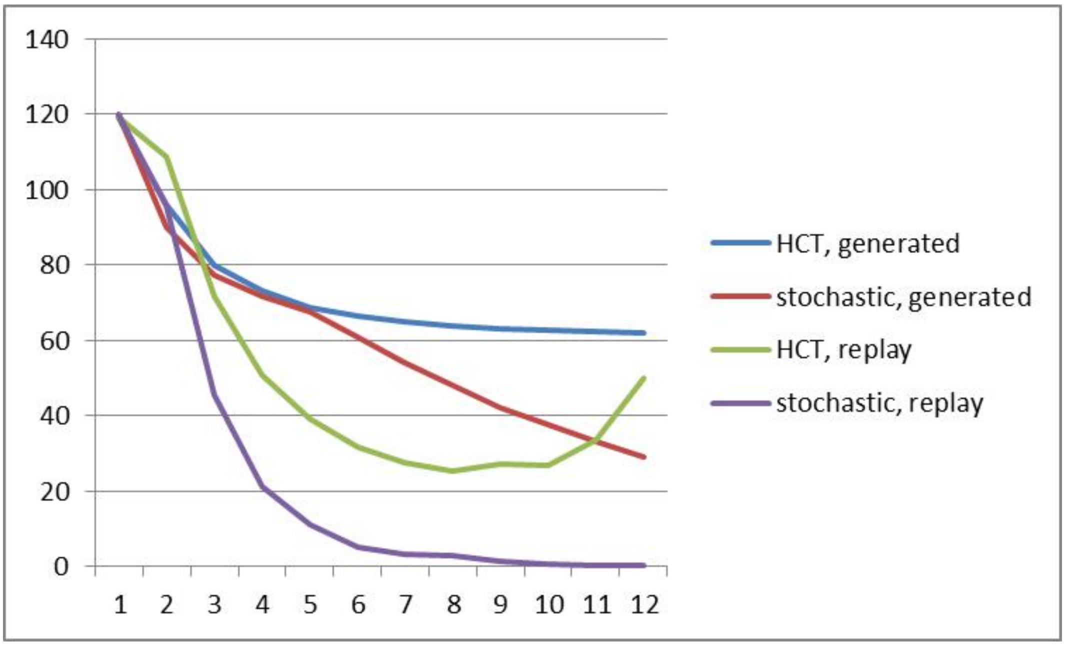

FIGURE 4.5: Average number of considered decks of 'Paladin' using the stochastic method and HCT methods with generated card sequences and sequences from replay data

for the stochastic method. After turn ten, the number of the considered card in the HCT method with the replay card sequence rises drastically.

Figure 4.5 shows the average number of considered decks per turn using the stochastic method and the HCT method with generated sequences of cards and replay sequences.

The number of considered decks is decreasing for both card prediction methods and card sequences. Both methods predict approximately the same number of cards per turn for the generated card sequence until turn five. The number of predicted cards is slightly decreasing with progressing turns. While the number of decks is stabilizing for HCT method, the number continues to decrease for the stochastic method.

In contradiction to the curves of the generated card sequences, the number of considered decks is decreasing more strongly for the replay card sequences. While the number of the stochastic method tends to zero, it is raising for the HCT method after turn ten.

4.2.3 Card Prediction Validation

The success rate of each in Subsection 4.2.2 made card predictions is validated. For the prediction using the HCT method three different thresholds are chosen, $T_1 = 0.75$, $T_2 = 0.9$, $T_3 = 0.99$. For each card sequence, at the start of each turn, the algorithm is asked what would be the top ten card that would be played in this turn. Then, the prediction is compared to the effectually played cards of this turn. The prediction is marked as correct for the rank x if at least one of the effectually played cards is

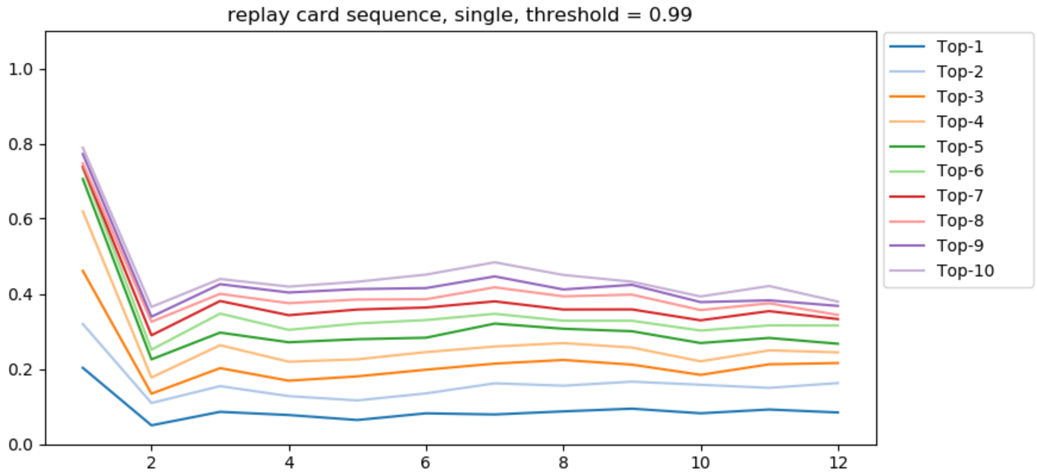

FIGURE 4.6: Success rate of the top ten card predictions

among this top x most probable card suggestions. The ranks $< x$ cards are marked as incorrect if the card is the x-th most probable card. The success rate for the top x prediction for a turn is calculated by the number marks for this turn divided by the number of sequences that contained data for this turn, i.e., the game was at least such many turns long. Figure 4.6 shows the success rate of the top ten card prediction of the HCT which was clustered using single linkage for the replay sequences of cards based on a threshold of 0.99. It is the average over all heroes. The Figure D.1 show the success rates of remaining combinations of prediction method and used card sequence. Figure D.2 shows the relation of the success rate of the stochastic method and HCT method with the thresholds 0.99, 0.9 and 0.75.

It can be seen, that the success rate is decreasing in the first turns and then increasing in the following turns in generated card sequences. Over all turns a high success rate can be achieved. In various ranking plots of generated card sequences, the stochastic method has similar success rate as the HCT method. In replay card sequence, the success rate is dropping after turn one, than increasing slightly, followed by stagnation for the HCT method, decreasing for the stochastic method. The HCT method has a higher success rate than the stochastic method in plots of card sequences from replays. For the higher ranks, e.g. top ten, it is greater than for the lower ranks. e.g. top 1, since it the accumulation of the previous ranks. The increase of the success rate is smaller with increasing rank. The success rate is different by each linkage methods, though, it distinguishes not significantly. Among the HCT method plots, the algorithm with the highest threshold performed best.

4.2.4 Discussion

The observations made in Subsection 4.2.1 regarding the course of the curves of the number of considered cards can be explained by the source of the card sequences. While cards of the metagame are played in the generated card sequence, cards not contained in the metagame are played in the replay card sequences which causes the deviation in the curve behavior. The HCT method tries to solve this problem by adding decks and cards into consideration, represented by a firstly stagnating graph, i.e., the same number of cards is removed and added, and a rising graph later on. Whereby, the stochastic method removes decks and their cards from consideration which do not contain the card of the card sequence. The number of considered cards is decreasing more slightly than the number of considered decks since the remaining decks tend to share the same cards. The only commonality of all graphs with regards to the number of considered cards which is increasing in the first turns is due to the increasing amount of Mana.

The first observation made from the validation in 4.2.3 is that the function worked as intended. The as most probable identified card, top one, has the highest success rate. Each next probable card has a lower success rate than the previous one. The distance of graphs visualizes the success rate of the cards. Since the top tenth graph has the smallest distance to a previous graph, the tenth most probable card has the smallest success rate. All predictions have their lowest success rate in turn four.

Like predicted, the number of correct card predictions is decreasing for the first turns, due to the increasing number of possible cards each turn. Cards cannot be excluded from the consideration since the already seen low Mana cards are played in nearly every deck. The number of the cards which are added to the consideration is increasing more strongly than the already seen cards are excluding cards from consideration, resulting in the descending graph. After turn four, the success rate is increasing for the generated card sequence as predicted, since the already played cards are marking specific subclusters more likely than others. Consequentially, the unlikely cluster is not considered further on, and the cards are removed from consideration. Furthermore, the number of playable cards has a smaller effect than before, since it is growing slower than the number of removed cards.

Under Consideration of the success rate, the stochastic method performs as well as the HCT method in card prediction using generated card sequences. On the contrary, the performance of it is weaker in the card prediction using card sequences from replays. It can be explained by the problems the stochastic method has which are described in Subsection 3.3. The card sequences from the replays, cards are played which are not in the decks of the metagame. It can not be excluded, that these cards are added to a deck by card effects. Though the replays were filtered beforehand, cards without the filtered keywords might exist which create unknown cards in the deck or hand. Though the HCT method performs better than the stochastic method using the card sequences from replays, the prediction could not be met regarding this sequences.

Observations which can be made on the different thresholds yield that a sharp threshold, i.e., the algorithm is less likely to select a subcluster and remove cards from consideration, is more suitable for the card prediction than a small one for the first turns. That is represented by the higher success of the sharp threshold. Especially, since in the first turn of a small threshold, the success rate of the HCT for the lower ranks is zero. Since few cards are already played in the first turns, the algorithm tends to select a subcluster due to the small threshold.

Chapter 5

Conclusion

In the course of this work, properties and special features on the subject of collectible card games, in particular, Hearthstone, were analyzed. The core characteristic was the variance of the possible decks/strategies, which made it possible to create one's deck out of a set of possible cards. Since multiple instances of the same card can appear in a deck and their order is irrelevant, as a result, decks were defined as multisets of cards. The metagame was identified to be the preference of players to play particular kinds of decks, and it was suggested as a way to help determine enemy moves. It includes the most played decks, the number of their plays and the win rate against other decks. From the number of games of preferred decks and their cards, the general probabilities for cards can be calculated. From already played cards, the probabilities for decks and the dependent probabilities of cards can be determined. It was recognized that a purely stochastic procedure could not deal with the problem of decks and cards which do not appear in the metagame, which can nevertheless be played in games. As a solution a clustering procedure was proposed, from which the hierarchical procedure was identified as the best implementation. A particular achievement of the analysis was the development of a Hierarchical Cluster Tree (HCT), the tree, which is created by the cluster steps and used to predict cards and decks. The clustering was validated by comparison of clusters with class labels which were found in the metadata and created by professional players. Different distance functions and linkage methods were examined, at which point the combination Jaccard distance and single linkage turned out to be the best one.

The card prediction using the HCT was verified with all linkage methods, even though single linkage proved to be the best clustering method. Another factor taken into account was the threshold used when traversing the HCT to remove unlikely decks from the consideration. Using card sequences created from the metagame on the one hand and card sequences replay data on the other, an it was evaluated how the predicted cards relate to the effectually played cards. Furthermore, the stochastic calculation method was validated with the help of these card sequences for comparison purposes. It turned out that the HCT and stochastic method achieved approximately the same performance with generated card sequences that do contain only cards included in the metagame. Using the real card sequences from replay data showed that the HCT method performed better. Furthermore, a high threshold for the HCT method has generally proven to be better, i.e., considering unlikely decks and ignoring high unlikely decks. Cards from generated sequences that were among the most likely ten were predicted with 80% accuracy using the HCT. Predicted cards from replay sequences, on the other hand, were only 40% accurate. The different sources of meta and replay data and the effects of individual cards to add more cards to the deck were identified as possible explanations. The former can be intercepted by the independent acquisition of data and the latter by an operating agent which enables the handling of certain card effects. The goals of this thesis could only be achieved for the generated map sequences.

From the results obtained, further investigations into this procedure are suggested such as the for the improvement of the performance of the HCT method through the integration of the suggested solutions. It would be essential for the independent game recording to know how much information is available to the recording agents.

- If he receives the cards at least after the game, he would be able to represent a complete metagame.
- If the information received would be limited to the cards played, the reconstruction of the metagame would be conceivable.

A distinction would have to be made between what performs better and what is possible.

An agent could detect unwanted card effects which are interfering with card prediction. This agent would make it possible to use the developed HCT method in real games instead of generated and replay card sequences. It would be necessary to investigate how the card prediction performs in actual games. Two types of use are thinkable.

- The use is as an aid for players and especially new players to identify dangers and modes of operation of enemy decks, especially concerning archetypes, i.e., the cluster. A user study could reveal how well users would receive such a decision support system.
- The use as card prediction in an agent which actually plays the game. It can use it to simulate possible future games states. With the in Subsection 2.5.2 presented game state evaluation, it would be able to determine its next move.

Apart from the use of the HCT card prediction in CCG, the performance in other fields of application which develop a metagame or something alike could be examined. This field must be similarly structured as the cards and decks. Such as, Scientific publications are tagged with specific keywords to their subject. These tags function as the card in the decks. They could find publications which contain the tags. In contrast to a search with a filter, similar publications and fields could be found. A disadvantage would be the favoring of clusters with a high number of data points by the algorithm, since fields which are not in focus of scientific research, though share many of the tags, would be most likely ignored due to their small number of publications. However, this could be solved by an intelligent weighting method of the nodes in the tree. So, for instance, rather than weighted by the number of publications which share the same keyword, the nodes could be weighted by the number of citations or number of views. Furthermore, the goal of the search could be the keyword instead of the publications they are tagging. The keyword ’linkage’ for example could be found if it is searched for words ’single’, ’complete’ and ’ward’. Similar to publications, an automatic detection of emerging genres could advance multimedia retrieval systems such as films, music, and games. As well as content recommendation, when ’deck’ of the user contains e.g. the videos it liked. How the algorithm performs in these fields of application, has to be investigated.

List of Figures

List of Tables

List of Abbreviations & Symbols

CCG	**C**ollectible **C**ard **G**ame
CI	**C**omputational **I**ntelligence
HCT	**H**ierachical **C**luster **T**ree
$\mathcal{C}$	Combination
C	set of clusters
c	cluster, multiset of decks
D	set of all decks
d	deck, multiset of cards with $s(d) = 30$
dis	distance
dis_{Manh}	Manhatten distance
dis_{Eucl}	Euclidean distance
dis_{Jac}	Jaccard distance
γ	count of an element in a multiset
K	set of all cards
$\mathcal{K}$	multiset of cards
k	card
M	multiset / metagame, multiset of decks
P	Permutation
Pr	Probability
s	size, number of element in a multiset
V	Variation
Υ_ι	multiset of cards
ι	count of a cards in a deck

Appendix A

Distance Matrices

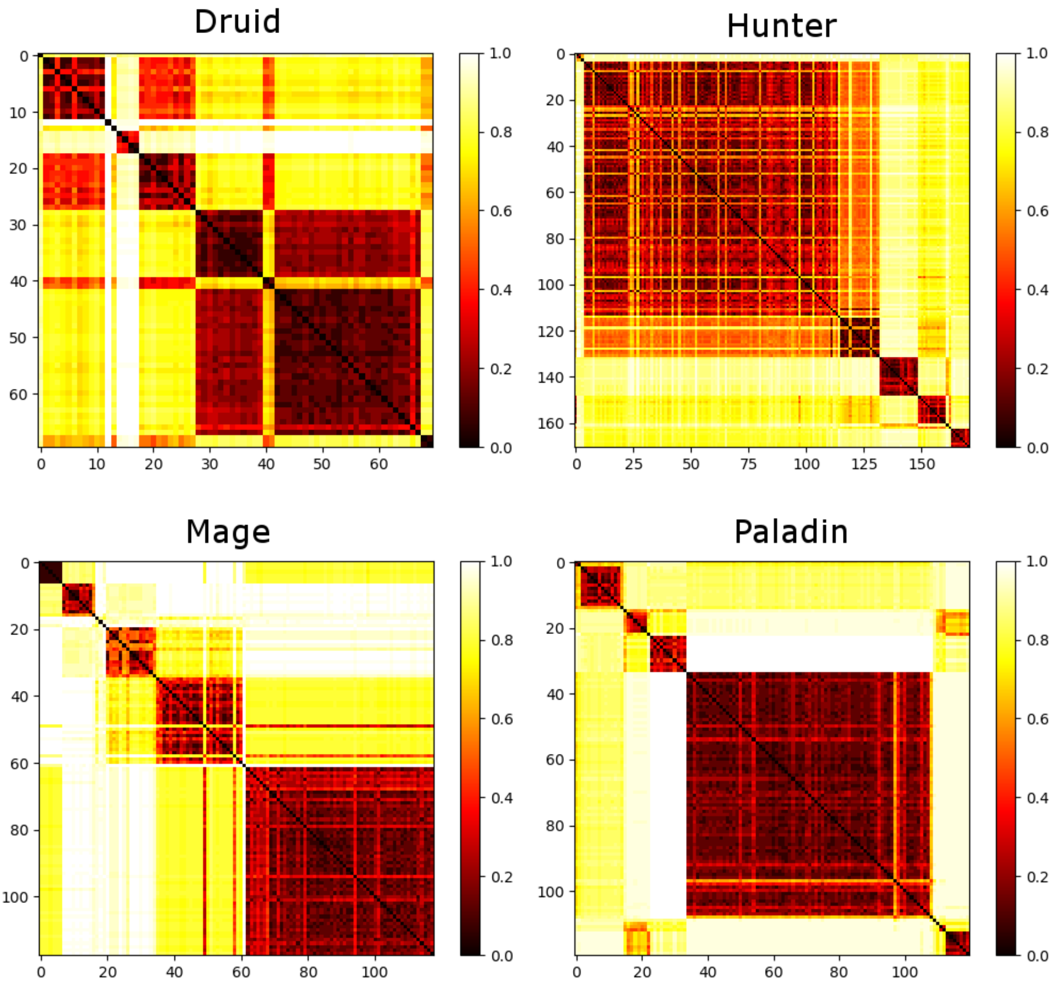

FIGURE A.1: Jaccard distance matrices of the decks 'Druid', 'Hunter', 'Mage', and 'Paladin'

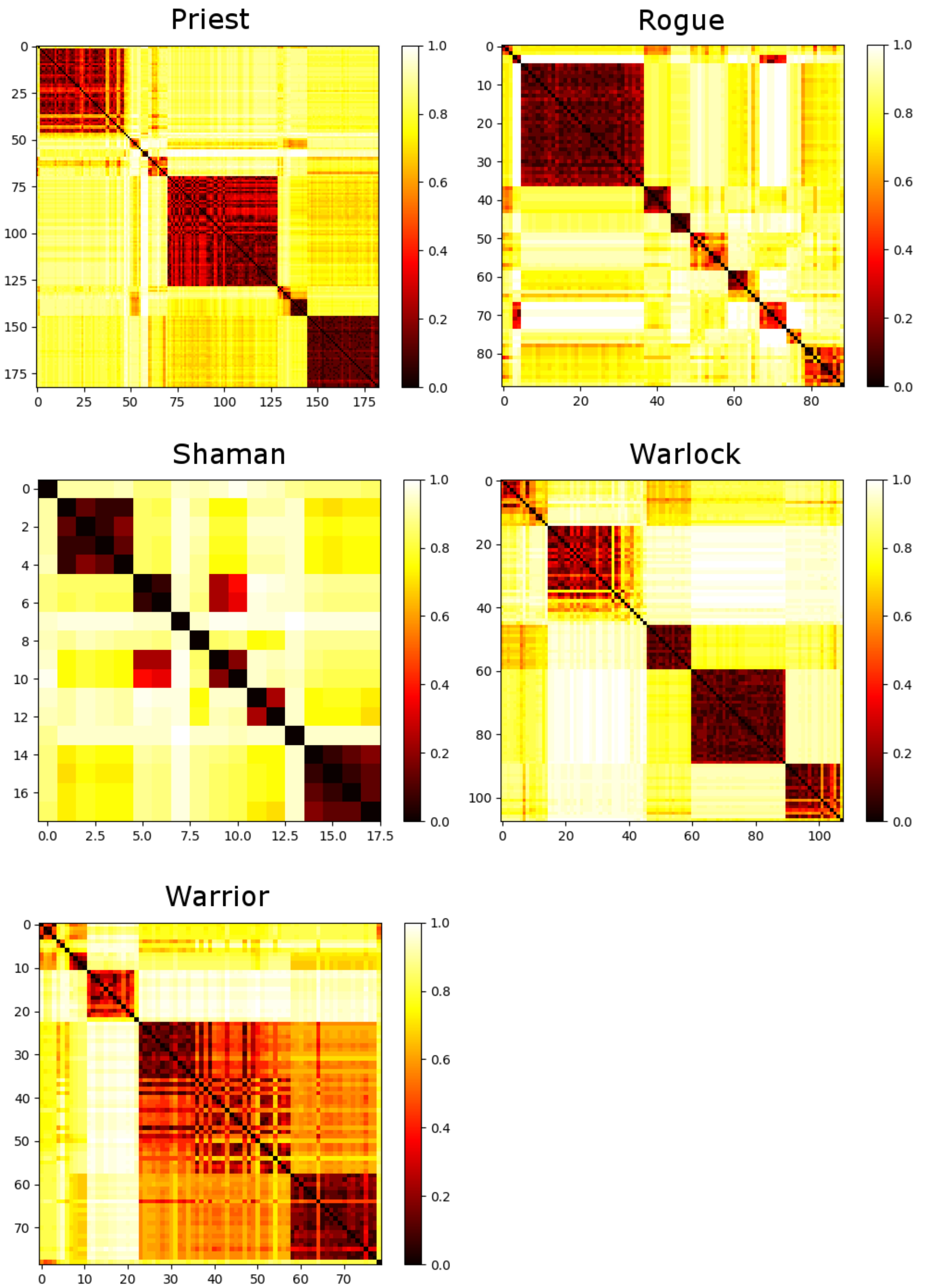

FIGURE A.2: Jaccard distance matrices of the decks 'Priest', 'Rogue', 'Shaman', 'Warlock', and 'Warrior'

Appendix B

Dendrograms

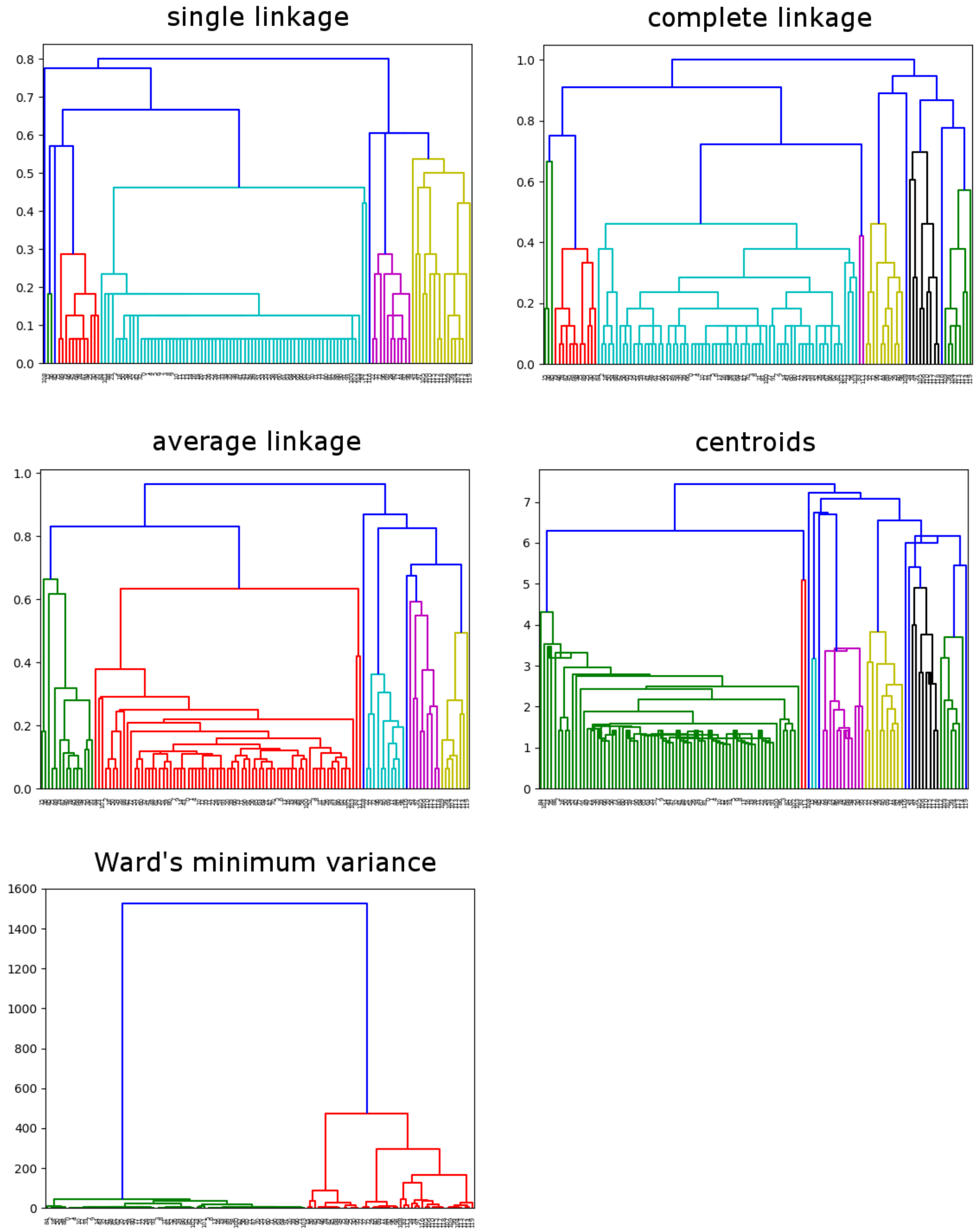

FIGURE B.1: Dendrograms of 'Paladin' decks, clustered by Jaccard distance and linkage methods, single linkage, complete linkage, average linkage, centroids and Ward's minimum variance

Appendix C

V-measure

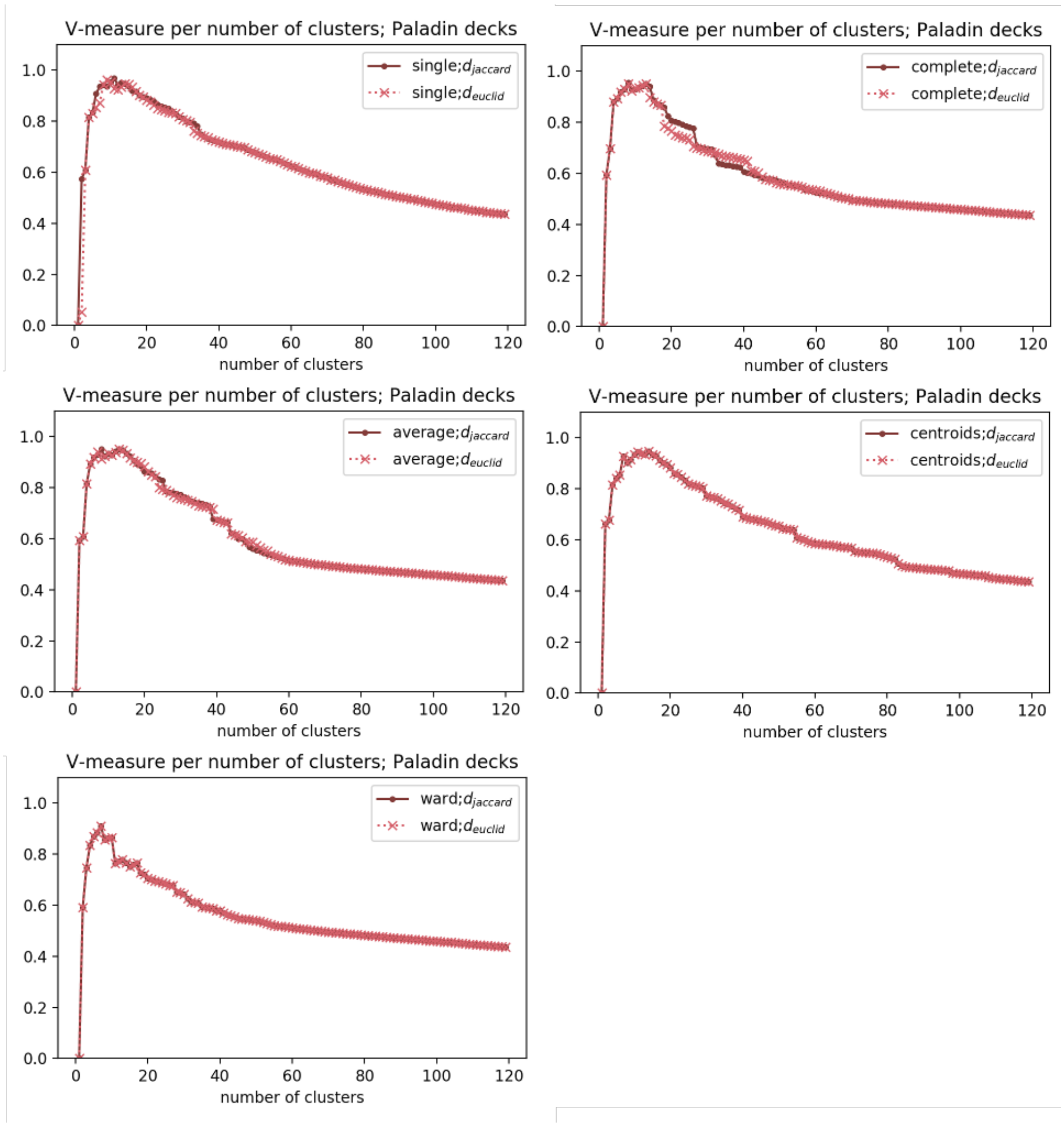

FIGURE C.1: V-measure of 'Paladin' decks, clustered by Jaccard distance and linkage methods, single linkage, complete linkage, average linkage, centroids and Ward's minimum variance

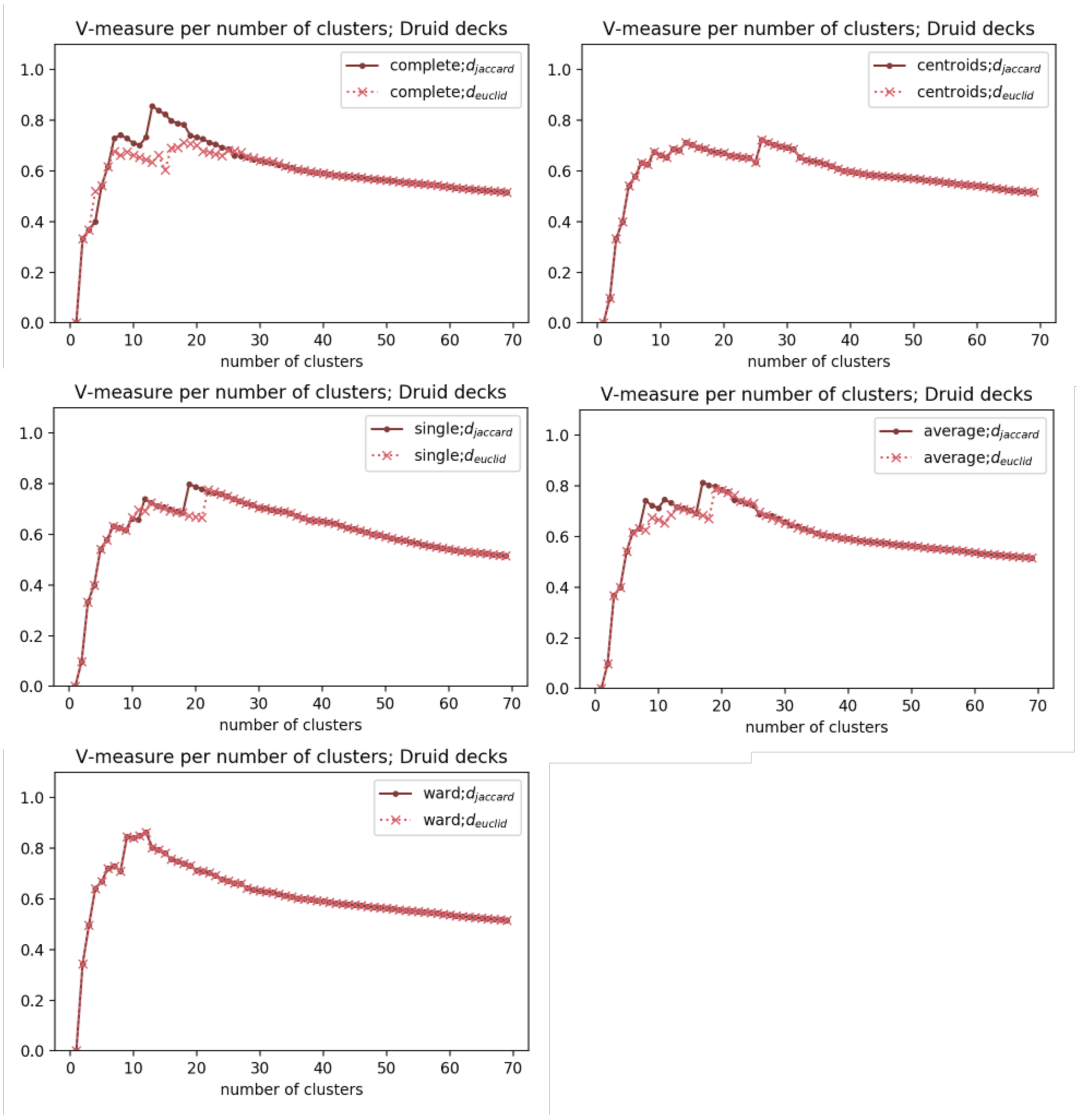

FIGURE C.2: V-measure of 'Druid' decks, clustered by Jaccard distance and linkage methods, single linkage, complete linkage, average linkage, centroids and Ward's minimum variance

Appendix D

Success Rate of Card Predictions

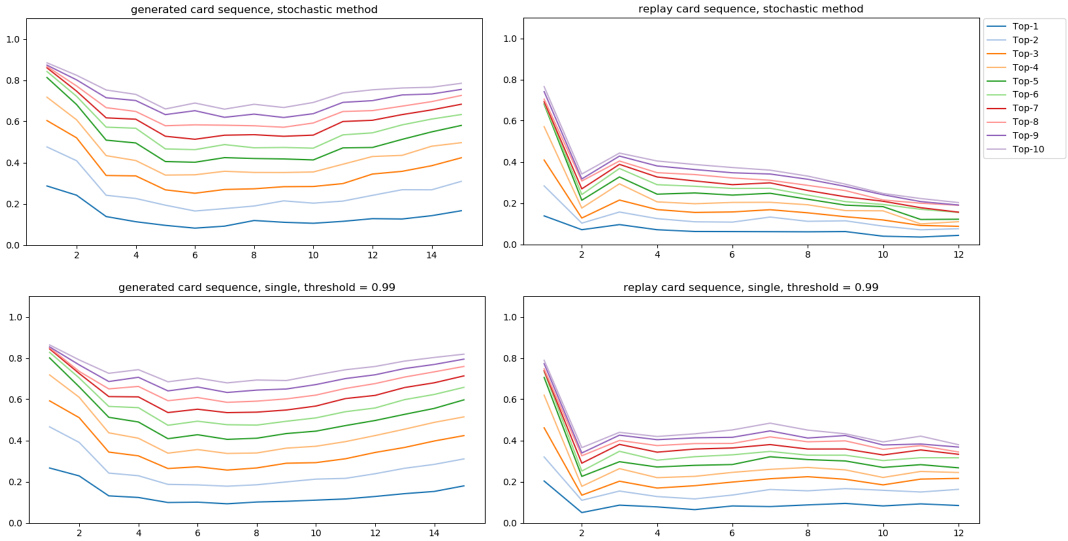

FIGURE D.1: Average success rate of the top 10 predictions using the combinations of generated and replay card sequences, and stochastic method and HCT method with a threshold of 0.99, HCT is clustered by Jaccard distance and single linkage

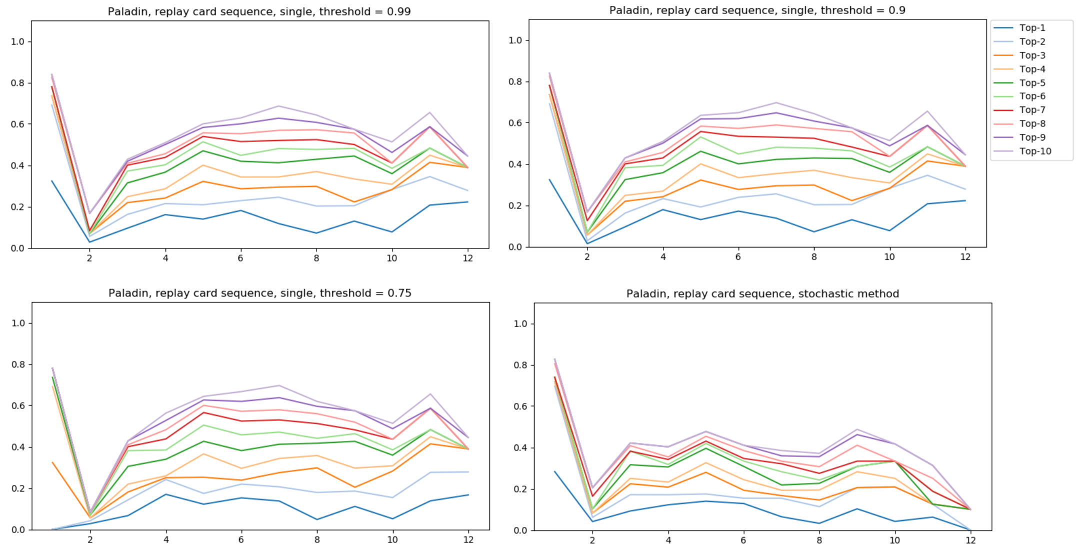

FIGURE D.2: Success rate of the top 10 predictions of the 'Paladin' deck using the combination of replay card sequences, and stochastic method and HCT method with a threshold of 0.99, 0.9 and 0.75, HCT is clustered by Jaccard distance and single linkage

Bibliography

[1] P. McCorduck. *Machines Who Think: A Personal Inquiry into the History and Prospects of Artificial Intelligence*. A K Peters, 2004.

[2] V. Akman A. P. Saygin I. Cicekli. "Turing Test: 50 Years Later". In: *Minds and Machines* 10.4 (Nov. 2000), pp. 463–518.

[3] University of Washington. *The History of Artificial Intelligence*. 2006. URL: https://courses.cs.washington.edu/courses/csep590/06au/projects/history-ai.pdf.

[4] F. Hsu M. Campbell A. J. Hoane Jr. "Deep Blue". In: *Artificial Intelligence* 134.1-2 (Jan. 2002), pp. 57–83.

[5] *Intelligente Roboter kommen nach Europa*. Feb. 2016. URL: https://www.n-tv.de/wirtschaft/Intelligente-Roboter-kommen-nach-Europa-article17082191.html.

[6] P. Jonker L. d. Witte R. Bemelmans G. J. Gelderblom. "Effectiveness of Robot Paro in Intramural Psychogeriatric Care: A Multicenter Quasi-Experimental Study". In: *jamda* 16.11 (Nov. 2015), pp. 46–50. URL: https://www.jamda.com/article/S1525-8610(15)00345-X/pdf.

[7] L. Aronson. *The Future of Robot Caregivers*. July 2014. URL: https://www.nytimes.com/2014/07/20/opinion/sunday/the-future-of-robot-caregivers.html.

[8] *Mars Fact Sheet*. URL: https://nssdc.gsfc.nasa.gov/planetary/factsheet/marsfact.html.

[9] M. Koren. *The Mars Robot Making Decisions on Its Own*. June 2017. URL: https://www.theatlantic.com/technology/archive/2017/06/mars-curiosity-rover/531339/.

[10] P. Brasor. *Shogi: A measure of artificial intelligence*. June 2017. URL: https://www.japantimes.co.jp/news/2017/07/08/national/media-national/shogi-measure-artificial-intelligence/#.W8Xw5flCSpo.

[11] E. Zimmermann K. Salen. *Rules of Play - Game Design Fundamentals*. 1st ed. The MIT Press, 2004. ISBN: 0-262-24045-9.

[12] *Magic the Gathering*. 1993-2019. URL: https://magic.wizards.com/.

[13] *Pokemon Traiding Card Game*. 1996-2019. URL: https://www.pokemon.com/us/pokemon-tcg/play-online/.

[14] *Yu-Gi-Oh!* 1996-2019. URL: `https://www.yugioh.com/`.

[15] *Hearthstone*. 2014-2019. URL: `https://playhearthstone.com/`.

[16] *Shadowverse*. 2016-2019. URL: `https://shadowverse.com/`.

[17] *Hearthstone Wiki*. written and maintained by players. 2019. URL: `https://hearthstone.gamepedia.com/`.

[18] *HSReplay.net*. URL: `https://hsreplay.net/`.

[19] J. Schell. *The Art of Game Design – A Book of Lenses*. Elsevier Inc., 2008. ISBN: 978-0-12-369496-6.

[20] Stanislav Costiuc. *What Is A Meta-Game?* Feb. 2019. URL: `https://www.patreon.com/posts/farlands-what-is-24635737`.

[21] J. Anderson G. W. Arnold F. Canavero M. El-Hawary B-M. Haemmerli M. Lanzerotti D. Jabobson O. P. Malik S. Nahavandi T. Samad G. Zobrist L. Hanzo R. Abhari. *DATA MINING - Concepts, Models, Methods, and Algorithms*. 2nd ed. IEEE Press, John Wiley & Sons, 2011. ISBN: 978-1-118-02912-1.

[22] J. Togelius G. N. Yannakakis. *Artificial Intelliegence and Games*. 1st ed. Springer, Jan. 2018. ISBN: 978-3-319-63519-4.

[23] A. Rana A. Singh A. Yadav. "K-means with Three different Distance Metrics". In: *International Journal of Computer Application* 67.10 (Apr. 2013), pp. 13–17.

[24] D. Winter M. Levandowsky. "Distance between Sets". In: *Nature* 234 (Nov. 1971), pp. 34–35.

[25] R. Tibshirani J. Bien. "Hierarchical Clustering With Prototypes via Minimax Linkage". In: *Journal of the American Statistical Association* 106.495 (2011), pp. 1075–1084. DOI: `10.1198/jasa.2011.tm10183`. eprint: `https://doi.org/10.1198/jasa.2011.tm10183`. URL: `https://doi.org/10.1198/jasa.2011.tm10183`.

[26] D. Winter M. Levandowsky. "Hierarchical Clustering via Joint Between-Within Distances: Extending Ward's Minimum Variance Method". In: *Journal of Classification* 22.2 (Sept. 2005), 151–183.

[27] *Mathworks-Dendrogram*. 2019. URL: `https://www.mathworks.com/help/stats/dendrogram.html`.

[28] H. Rinne. *Taschenbuch der Statistik*. 4th ed. Verlag Harri Deutsch, 2008. ISBN: 978-3-8171-1827-4.

[29] S. Schäffler D. Maintrup. *Stochastik : Theorie und Anwendungen*. Springer, 2005.

[30] S. Miyamoto. "Fuzzy multisets and their generalizations". In: *Multiset processing. Mathematical, computer science, and molecular computing points of view*. 2000, pp. 225–236.

[31] D. E. Knut. *Seminumerical Algorithms*. Vol. 2. Addison-Wesley, 1969.

[32] R. R. Yager. "On the theory of bags". In: *International Journal of General Systems* 13 (1986), pp. 23–37.

[33] R. Kruse A. Dockhorn T. Schwensfeier. "Fuzzy Multiset Clustering for Metagame Analysis". In: *Proceedings of the 11th EUSFLAT Conference*. submitted. 2019.

[34] Jan Jakubik. "Evaluation of Hearthstone Game States With Neural Networks and Sparse Autoencoding". In: *Proceedings of the Federated Conference onComputerScience and Information Systems*. Vol. 11. 2017, pp. 135–138.

[35] V. Akman A. P. Saygin I. Cicekli. "Hearthstone: Finding the optimal play". MA thesis. Ghent University, 2018.

[36] E. Bursztein. "I am a legend: Hacking Hearthstone". In: *DEFCONConference*. Vol. 22. 2015.

[37] Flipperbw. *Simple hearthstone logging - see your complete play history without TCP, screen capture, or violating the TOS*. 2014. URL: https://www.reddit.com/r/hearthstone/comments/268fkk/simplehearthstoneloggingseeyourcompletepla.

[38] *Leading digital collectible card game (CCG) titles worldwide in 2019, by follower base on streaming platform twitch*. data collected at 16.02.2019 10:00(CET). 2019. URL: https://www.twitch.tv/.

[39] Statista. *Leading digital collectible card game (CCG) titles worldwide in 2016, by revenue*. 2019. URL: https://www.statista.com/statistics/666594/digital-collectible-card-games-by-revenue/.

[40] *Sabberstone*. URL: https://hearthsim.info/sabberstone/.

[41] N. M. Thalmann Q. Yuan G. Cong. "Enhancing naive bayes with various smoothing methods for short text classification". In: *Proceedings of the 21st International Conference on World Wide Web*. ACM. 2012, pp. 645–646.

[42] Alexander Dockhorn, Christian Braune, and Rudolf Kruse. "Variable density based clustering". In: *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, Dec. 2016, pp. 1–8. ISBN: 978-1-5090-4240-1. DOI: 10.1109/SSCI.2016.7849925. URL: http://ieeexplore.ieee.org/document/7849925/.

[43] Alexander Dockhorn, Christian Braune, and Rudolf Kruse. "An alternating optimization approach based on hierarchical adaptations of dbscan". In: *2015 IEEE Symposium Series on Computational Intelligence*. IEEE. 2015, pp. 749–755.

[44] J. Hirschberg A. Rosenberg. "V-Measure: A Conditional Entropy-Based External Cluster Evaluation Measure". In: *EMNLP-CoNLL 2007, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, June 28-30, 2007, Prague, Czech Republic*. 2007, pp. 410–420.

[45] *Track-o-Bot*. 2019. URL: https://trackobot.com/.

[46] *Collect-o-Bot*. 2019. URL: http://www.hearthscry.com/CollectOBot.