Modeling and Analyzing Social Networks of Games

Leonardo Mauro Pereira Moraes
Dissertação de Mestrado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)
Leonardo Mauro Pereira Moraes

Modeling and Analyzing Social Networks of Games

Master dissertation submitted to the Institute of Mathematics and Computer Sciences – ICMC-USP, in partial fulfillment of the requirements for the degree of the Master Program in Computer Science and Computational Mathematics. FINAL VERSION

Concentration Area: Computer Science and Computational Mathematics

Advisor: PhD. Robson Leonardo Ferreira Cordeiro

USP – São Carlos
June 2020
Moraes, Leonardo Mauro Pereira
Modeling and analyzing Social Networks of Games
/ Leonardo Mauro Pereira Moraes; orientador Robson Leonardo Ferreira Cordeiro. -- São Carlos, 2020.
110 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2020.

1. Social Network. 2. Player Modeling. 3. Data Mining. 4. Digital Influencers. 5. Game Design. I. Cordeiro, Robson Leonardo Ferreira, orient. II. Título.
Leonardo Mauro Pereira Moraes

Modelando e Analizando Redes Sociais de Jogos

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências – Ciências de Computação e Matemática Computacional. VERSÃO REVISADA
Área de Concentração: Ciências de Computação e Matemática Computacional
Orientador: PhD. Robson Leonardo Ferreira Cordeiro

USP – São Carlos
Junho de 2020
Everyone is connected with some technological device. It is visible the fascinating things that it proposes to us: from a long-distance video chat to a moment of play with friends.

The Master System taught me Alex Kidd’s perseverance and showed the incredible scenarios of Sonic Hedgehog; GameBoy Color, I can’t forget those 151 pokemon that gave me so much delight in capturing.

I dedicate this work for the enthusiasts of games and technologies, as well as for the creators of these unforgettable games.
I’m grateful to my parents, and especially to my grandfather, for supporting me and believing me. In addition to the most important person for me, my wife, for everything.

I thank my advisor for teaching me so much in a short time, as well as my colleagues in the laboratory. In addition to all the university for the help and its dedicated professors.

Besides, I am grateful for the support of the Brazilian National Council for Scientific and Technological Development (CNPq); Coordination for the Improvement of Higher Education Personnel - Brazil (CAPES) [grant 001]; Sao Paulo Research Foundation (FAPESP) [grant 2018/05714-5]; and AWS Cloud Credits for Research.
“Don’t use force in training the children in the studies, but rather play. In that way you can better discern what each is naturally directed towards.”

(Plato)
ABSTRACT

MORAES, L. M. P. Modeling and Analyzing Social Networks of Games. 2020. 110 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2020.

Digital games are dynamic environments where the players interact with the games and, commonly, also with other players. The players’ interactions have many types of relationships, both with other players (e.g., be friend, share games) and with games (e.g., buy, play, like); these relationships can be represented by social networks, i.e., in this context, a Social Network of Games (SNG). During a gameplay, the player produces a vast amount of data continuously: data that represents his/her experiences, preferences and behavioral patterns. In this way, it is possible to use these data to understand the players’ preferences, i.e., Player Modeling, and to improve the attractiveness of new games. However, this task requires the analysis of a vast amount of data, what is impracticable to do manually, then requiring robust algorithms of a research area named Knowledge Discovery in Databases (KDD) to overcome such an issue. It is possible to apply KDD techniques in data of a SNG to model the players, as well as to identify intrinsic features of the games. This MSc work aimed to explore a real SNG using KDD techniques for identifying common features among popular games, which may represent reasons why the games are popular. However, to achieve this goal, it is necessary to analyze games developed by non-influencer makers, because influencers may receive biased attention in their games that is not necessarily motivated by the game quality. First, we focused on detecting the game influencers to filter them out; empirical experiments show that our approach automatically detects influencers with high precision, even when using data from distinct nationalities for training and testing. Then, we performed a detailed analysis of games’ features, searching for object combinations that occur in the popular games developed by non-influencer makers, so to support game designers in the elaboration process of new games. This case study introduces a new design pattern for platform games. Also, we present an extensive analysis of object combinations that commonly occur in popular games. All experiments were performed on players and games from the worldwide well-known Super Mario Maker (Nintendo, Kyoto, Japan).

Keywords: Social Network of Games, Player Modeling, Data Mining, Detection of Influencers, Game Design.
Os jogos digitais são ambientes dinâmicos em que o jogador interage com o jogo e, muitas vezes, também com outros jogadores. As interações dos jogadores representam diversos tipos de relacionamentos, seja com outros jogadores (e.g., ser amigo, compartilhar jogos) ou com os jogos (e.g., comprar, jogar, curtir); estes relacionamentos podem ser representados por redes sociais, neste contexto, uma Redes Social de Jogos (SNG). Durante suas partidas, o jogador produz uma grande quantidade de dados continuamente: dados que representam as suas experiências, preferências e padrões comportamentais. Desta forma, é possível utilizar tais dados para compreender as preferências dos jogadores, i.e., Modelagem de Jogador, e para tornar novos jogos mais atrativos. Contudo, isto requer a análise de uma grande quantidade de dados, uma tarefa impraticável de se fazer manualmente, e assim faz-se necessário o uso de algoritmos robustos de uma linha de pesquisa denominada Descoberta de Conhecimento em Base de Dados (KDD). É possível utilizar técnicas de KDD em dados de uma SNG para modelar os jogadores, bem como identificar características intrínsecas dos jogos. O presente trabalho de Mestrado visa explorar uma SNG real utilizando técnicas de KDD com o propósito de identificar características comuns em jogos populares, as quais podem ser causas para a sua popularidade. Porém, para alcançar este objetivo, é necessário analisar jogos elaborados por desenvolvedores não-influenciadores, pois os influenciadores, por sua vez, podem receber atenção em seus jogos pormotivos alheios à qualidade do jogo. Inicialmente, nos concentramos em detectar os influenciadores para filtrá-los; experimentos empíricos demonstraram que o nosso método detecta influenciadores automaticamente com alta precisão, mesmo quanto foram utilizados dados de países distintos para treinamento e teste. Em sequência, realizamos uma análise detalhada sobre as características dos jogos, em busca de combinações de objetos que que são comuns em jogos populares elaborados por usuários não-influenciadores, com o objetivo de auxiliar o profissional no papel de designer de jogos a elaborar novos jogos. Este estudo de caso propõe um novo padrão de design para jogos de plataforma, bem como apresenta uma extensa análise sobre as combinações de objetos comumente presentes em jogos populares. Todos os experimentos foram realizados sobre jogadores e jogos do Super Mario Maker (Nintendo, Quioto, Japão), o qual é bem conhecido mundialmente.

Palavras-chave: Redes Social de Jogos, Modelagem de Jogador, Mineração de Dados, Detecção de Influenciadores Digitais, Design de Jogos.
LIST OF FIGURES

Figure 1 – Knowledge Discovery in Databases ........................................... 32
Figure 2 – Example of a directed graph ....................................................... 37
Figure 3 – Examples of different types of graphs ......................................... 37
Figure 4 – Example of a undirected graph .................................................. 39
Figure 5 – Relationship examples from Social Networks of Games ............... 46
Figure 6 – Types of player-game interactions on Super Mario Maker .......... 47
Figure 7 – Game styles in Super Mario Maker ........................................... 47
Figure 8 – SMM Bookmark: Game level .................................................... 48
Figure 9 – SMM Bookmark: Player-game interactions ................................... 48
Figure 10 – SMM Bookmark: Player’s statistics ......................................... 49
Figure 11 – Crawler: Data collection structure .......................................... 50
Figure 12 – Relational schema for SMMnet ............................................... 51
Figure 13 – Game level styles ................................................................. 52
Figure 14 – Game level difficulties ......................................................... 52
Figure 15 – The rank of game levels by number of player’ interactions for the top-100 most popular levels .......................................................... 53
Figure 16 – The probability density function for the top-100 most popular game levels with power-law and log-normal estimates ....................... 53
Figure 17 – The cumulative distribution function for the top-100 most popular levels with power-law and log-normal estimates ....................... 53
Figure 18 – The complementary cumulative distribution function for the top-100 most popular game levels with power-law and log-normal estimates ....... 53
Figure 19 – Game levels by country ......................................................... 54
Figure 20 – Social Networks $G_{dev}$ and $G_{like}$ changing over time .......... 58
Figure 21 – Stream modeling for the toy graph $G_{like}$ shown in Figure 20b .... 61
Figure 22 – Linear regression for a game level’s stream of like counts .......... 63
Figure 23 – Delta values $\Delta_{c,i}$ for a game $c$ ........................................ 65
Figure 24 – Coefficients of angle $\theta_{c,i}$ for a game $c$ .............................. 67
Figure 25 – Accuracy results for the top-6 best classifiers when using features from extractor LR. The input parameter $L$ was tuned with values from 1 to 10. .... 70
Figure 26 – Accuracy results for the top-6 best classifiers when using features from extractor DR. The input parameter $L$ was tuned with values from 1 to 10. .... 71
Figure 27 – Accuracy results for the top-6 best classifiers when using features from extractor CA. The input parameter $L$ was tuned with values from 1 to 10.

Figure 28 – Accuracy results for the top-6 best classifiers when using features from the ensemble of extractors $F_{ALL}$. The input parameter $L$ was tuned with values from 1 to 10.

Figure 29 – Our proposed framework to extract features from popular games.

Figure 30 – Our proposal for feature aggregation.

Figure 31 – Spotting subspace clusters.

Figure 32 – Pruning and analyzing.

Figure 33 – Head cut-off of clusters.

Figure 34 – Tail cut-off of clusters.

Figure 35 – Analyzing popular games with association rules.
Algorithm 1 – Extractor Linear Regression

Algorithm 2 – Extractor Delta Rank

Algorithm 3 – Extractor Coefficient of Angle
Table 1 – Datasets for player modeling .................................................. 45
Table 2 – Relation cardinality in SMMnet ................................................. 52
Table 3 – Game level styles ................................................................. 52
Table 4 – Game level difficulties ....................................................... 52
Table 5 – Game levels by country ...................................................... 54
Table 6 – Experiments: evaluation of the top-3 best classifiers in each experiment ..... 72
Table 7 – Number of clusters returned by CLIQUE versus the dimensionality of their subspaces ............................................................... 84
Table 8 – Top-25 clusters according to their sizes .................................. 85
Table 9 – Final set of clusters selected for analysis ................................... 86
Table 10 – Frequent itemsets sorted by their support ............................... 92
Table 11 – Association rules sorted by their confidence ............................ 93
LIST OF SYMBOLS

ACC — Accuracy metric
PRE — Precision metric
REC — Recall metric
$F_1$ — F-score metric
SUPP — Support metric
CONF — Confidence metric
LEV — Leverage metric
LIFT — Lift metric
CONV — Conviction metric
$G(V,E)$ — Graph $G$, or $G(V,E,\delta)$
$V$ — Nodes of the graph
$E$ — Edges of the graph
$K_n$ — Complete graph of $n$ nodes
$G_b$ — Bipartite graph, or bigraph
$G_c$ — Connected graph
$G_d$ — Disconnected graph
$n$ — Nodes counting
$m$ — Edges counting
$g(v)$ — Degree of the node $v$
$g_{in}(v)$ — Input degree of the node $v$
$g_{out}(v)$ — Output degree of the node $v$
$d(G)$ — Density of graph $G$
$s(G)$ — Sparsity of the graph $G$
## CONTENTS

1  INTRODUCTION ............................................................... 27  
   1.1 Motivation ............................................................... 27  
   1.2 Context ................................................................. 28  
   1.3 Problem Definition and Hypothesis .............................. 29  
   1.4 Main Contributions .................................................. 29  
   1.5 Conclusion ............................................................. 30  

2  FUNDAMENTAL CONCEPTS .................................................... 31  
   2.1 Knowledge Discovery in Databases ............................... 31  
       2.1.1 Data Mining ....................................................... 33  
          2.1.1.1 Classification ............................................... 33  
          2.1.1.2 Clustering .................................................... 34  
          2.1.1.3 Association Rules ......................................... 34  
   2.2 Player Modeling ...................................................... 35  
   2.3 Social Networks ...................................................... 36  
       2.3.1 Network Terminology ......................................... 36  
       2.3.2 Network Metrics ............................................... 38  
       2.3.3 Social Network Analysis ................................... 40  
   2.4 Discussion ............................................................. 41  

3  CRAWLING A SOCIAL NETWORK OF GAMES .......................... 43  
   3.1 Related Work .......................................................... 43  
   3.2 Social Network of Games ......................................... 45  
   3.3 Super Mario Maker ................................................... 46  
       3.3.1 Platform ........................................................... 46  
       3.3.2 SMM Bookmark .................................................. 47  
   3.4 SMMnet Dataset ....................................................... 48  
       3.4.1 Data Collection .................................................. 49  
       3.4.2 Data Description ............................................... 50  
       3.4.3 Data Characterization ....................................... 52  
   3.5 Applicability .......................................................... 54  
       3.5.1 Social Network Analysis ..................................... 54  
       3.5.2 Artificial Intelligence in Games ............................ 55
3.6 Discussion ................................................................. 55

4 DETECTING GAME INFLUENCERS ................................. 57
4.1 Network and Problem .................................................. 58
4.2 Related Work ............................................................. 59
4.3 Proposal ................................................................. 60
4.3.1 Stream Modeling ....................................................... 61
4.3.2 Extraction of Temporal Features ................................. 62
4.3.2.1 Extractor Linear Regression .................................... 62
4.3.2.2 Extractor Delta Rank ............................................... 64
4.3.2.3 Extractor Coefficient of Angle ............................... 65
4.3.2.4 Ensemble ............................................................ 67
4.4 Experimental Evaluation .............................................. 68
4.4.1 Datasets ............................................................... 69
4.4.2 Evaluation of Individual Feature Extractors .................... 70
4.4.3 Ensemble Experiment ............................................... 72
4.4.4 Validation ............................................................ 73
4.5 Discussion ............................................................... 74

5 CHARACTERIZING POPULAR GAMES ............................... 75
5.1 Related Work ............................................................. 76
5.2 Extracting Features from Popular Games .......................... 77
5.2.1 Player Filtering ......................................................... 78
5.2.2 Data Collection ........................................................ 79
5.2.3 Feature Extraction ..................................................... 79
5.2.4 Feature Aggregation .................................................. 80
5.3 Characterizing Popular Games ....................................... 82
5.3.1 Analyzing Popular Games with Subspace Clustering ........ 82
5.3.1.1 Spotting Clusters in Subspaces ............................... 83
5.3.1.2 Pruning Irrelevant Clusters ..................................... 84
5.3.1.3 Analyzing Clusters of Similar Game Levels ................ 88
5.3.2 Analyzing Popular Games with Association Rules ............ 91
5.3.2.1 Discovering Association Rules ................................. 91
5.3.2.2 Analyzing Association Rules ................................... 92
5.4 Discussion ............................................................... 94

6 CONCLUSION ............................................................ 97
6.1 Main Contributions ..................................................... 97
6.2 Future Work ............................................................ 98
6.3 Additional Contribution ............................................... 99
This chapter presents an overview of the MSc research. It contains brief descriptions of the facts that motivated this work, besides the problem definition, the main objectives and our contributions. The following sections describe each one of these topics.

1.1 Motivation

Game is a primitive concept that precedes the creation of our own culture, and refers to all kinds of ludic activities (Huizinga, 1955). Digital games or video games are electronic games developed to be used in technological devices (Pivec; Kearney, 2007; Yannakakis; Togelius, 2018). In spite of the fact that there are conceptual differences between games, digital games and video games, we use these terms interchangeably in this document.

Games have high applicability; they are widely used for entertainment as well as in education, arts, training and health, culture, museums, and scientific research (Yannakakis; Togelius, 2018). The game market is in constant expansion, both in production and in consumption. This industry moves billions of dollars per year (Lucas, 2009; Yannakakis; Togelius, 2018). According to Newzoo1, a specialist company of game marketing, the estimated market cap for 2019 is US$ 150 billion. Also, the popularization of games expands to professional competitions, TV shows and online streaming of game matches that are commonly known as gameplays (Yannakakis; Togelius, 2018). Only in the first quarter of 20192, streamers have produced more than 115.5 million hours of online videos related to gameplays on the platforms Twitch3 and YouTube3. Due to this popularization, the game companies invest constantly in the production of new games with increasing graphic and interactive quality (Yannakakis, 2012).

1 Newzoo. “Insights”. <https://newzoo.com/insights/articles/> (accessed June 17, 2019).
2 Twitch. <https://www.twitch.tv/> (accessed November 01, 2019).
3 YouTube. <https://www.youtube.com/> (accessed November 01, 2019).
Moreover, researchers use the simulated environment of games, in which several factors react simultaneously, to evaluate algorithms and propose novel adaptive approaches (YANNAKAKIS, 2012; YANNAKAKIS; TOGELIUS, 2015). Player Modeling (PM) is among the most prominent game research topics; it refers to the study of players’ experiences to understand their common preferences and behaviors (YANNAKAKIS; TOGELIUS, 2015; YANNAKAKIS; TOGELIUS, 2018). In the next section, we discuss this context further.

1.2 Context

The player-game interaction is complex because players usually perform many different actions in gameplays, thus producing massive amounts of data (YANNAKAKIS; TOGELIUS, 2015). This vast amount of data represents the players’ experiences (YANNAKAKIS; TOGELIUS, 2015; YANNAKAKIS; TOGELIUS, 2018). In this sense, researchers explore these data to understand the players’ experiences and to extract knowledge from them; this task is commonly known in the literature as Player Modeling (PM).

PM is a research area that infers a computational representation of players’ experiences; that is, it captures players’ features to understand their behavioral patterns. Therefore, PM extracts knowledge from data produced by players to study their interactions with games. This knowledge allows the identification of players’ preferences to be used in the development of more successful games (YANNAKAKIS et al., 2013).

**DEFINITION.** A successful game, or popular game, is one that attracts a large set of players and, respectively, receives many positive reactions, *e.g.* “likes”.

PM analyzes large amounts of data, impractical activity to do manually. Thus, automatic techniques from a research area named Knowledge Discovery in Databases (KDD) are therefore commonly employed. KDD can be seen as a set of tasks to extract or infer knowledge from a dataset of interest (KAMBER; HAN; PEI, 2012). In the case of PM, the dataset must represent interactions that occur in a platform of video games.

A platform of video games is an environment where players can interact with games (*e.g.*, buy, make, play, like) and, usually, also with other players (*e.g.*, be friend, share games) (JIA et al., 2015; EBERHARD et al., 2018). Provided that such interactions can be seen as a social network, this MSc work coins the term **Social Network of Games (SNG)** to refer to them. In this way, knowledge discovery in SNG allows us to understand patterns of players in a platform of video games, as well as to identify games’ features (LEE; LEE; CHOI, 2012; YANNAKAKIS et al., 2013).
1.3 Problem Definition and Hypothesis

As aforementioned, it is possible to discover knowledge from a SNG in order to understand patterns of players and game features in a platform of video games. With that in mind, this MSc work aimed to explore a real SNG using KDD techniques for identifying common characteristics among popular games, which may represent reasons why the games are popular. However, achieving this goal requires to analyze games developed by non-influencer makers, because influencers may receive biased attention in their games that is not necessarily motivated by the game quality. Thus, we have explored the following hypothesis:

*It is possible to use information from a Social Network of Games to support the development of new, more successful games by taking advantage of Knowledge Discovery in Databases to identify common characteristics present in popular games that were elaborated by non-influencer makers, so to avoid bias in the process.*

First, we focused on detecting game influencers to filter them out. Then, we performed a detailed analysis of games’ features, searching for object combinations that occur in popular games developed by non-influencer makers, to support game designers in the elaboration process of new games. This case study introduces a new design pattern for games of platform. Additionally, we present an extensive analysis of object combinations that commonly occur in popular games. All experiments were performed on players and games from the well-known Super Mario Maker (Nintendo, Kyoto, Japan).

1.4 Main Contributions

The main contributions of this MSc work are:

**C1. Social Network of Games:** we coin the term Social Network of Games and present a formal definition for it. We also describe the data collection process that was performed on the well-known Super Mario Maker, and report statistics on a public dataset generated by this process;

**C2. Detection of Game Influencers:** we propose a novel framework to detect game influencers in Social Network of Games. It includes a feature extractor to infer influencer features and a classification step to distinguish between influencer and non-influencer users. Empirical experiments show that our approach automatically detects game influencers with high precision, even when using data from distinct nationalities for training and testing;

**C3. Characterization of Popular Games:** we report the results of a detailed analysis of games’ features where we searched for object combinations that occur in popular games
developed by non-influencer makers, so to support game designers in the elaboration process of new games.

1.5 Conclusion

This chapter presented an overview of the MSc work with a brief description of its context, motivations and contributions. The rest of this monograph is structured as follows. Chapter 2 describes fundamental concepts that are the basis for our work. Chapter 3 presents a formal definition for Social Network of Games, and also describes the data collection process from the platform of games Super Mario Maker. Our proposal to detect game influencers is described in Chapter 4. Finally, Chapter 5 reports the proposed methodology to discover common characteristics of popular games, while conclusions are given in Chapter 6.
Digital games are developed for electronic devices, i.e., computers, mobile devices like cellphones and tablets, or even exclusive devices that are known as video game consoles. Games are one of the most significant entertainment fields (YANNAKAKIS; TOGELIUS, 2018); moreover, they are applied in education, simulation and health assistance (LUCAS, 2009; YANNAKAKIS; TOGELIUS, 2018). In this context, player-game interactions can be complex because they may vary from simple interactions generated with bi-dimensional controls up to complex ones, where a player interacts with many possibilities and commands simultaneously. Therefore, each game produces a vast and, potentially, infinite amount of data (YANNAKAKIS et al., 2013; YANNAKAKIS; TOGELIUS, 2015).

A large portion of these data represents preferences of players (YANNAKAKIS; TOGELIUS, 2015; YANNAKAKIS; TOGELIUS, 2018); which can be used to personalize and improve the attractiveness of a game (FUNGE, 2004; COPPIN, 2004; YANNAKAKIS et al., 2013). However, this is impracticable to be done manually, due to the large amount of data involved, then requiring robust algorithms from a research area known as Knowledge Discovery in Databases (KDD) to model and understand the experiences of players.

This chapter presents fundamental concepts related to the MSc work. Section 2.1 briefly describes the area of Knowledge Discovery in Databases, while the specific topic of Player Modeling is discussed in Section 2.2. Then, social network concepts are given in Section 2.3. Finally, Section 2.4 concludes the chapter.

2.1 Knowledge Discovery in Databases

This section discusses the process of Knowledge Discovery in Databases. At first, it is important to comment on what data, information and knowledge are. In the game context, data can be represented by raw values of player-game interaction, e.g., player $a$ attacked monster
Chapter 2. Fundamental Concepts

$b$ with strength $c$. It is usually stored in attributes that are also known as characteristics or features, *e.g.*, who attacked, who was attacked and hit force (KAMBER; HAN; PEI, 2012; GOLDSCHMIDT; PASSOS; BEZERRA, 2015). Information is the result of an interpretation of the data with meaning for predefined contexts, *e.g.*, player $a$ defeated monster $b$. At last, knowledge represents a set of patterns, rules or relationships extracted from pieces of information, *e.g.*, players that use balanced combinations of attacks with low and high strength are more likely to defeat monster $b$. Therefore, knowledge can be patterns or preferences of players (KAMBER; HAN; PEI, 2012; GOLDSCHMIDT; PASSOS; BEZERRA, 2015; YANNAKAKIS; TOGELIUS, 2018), among many other possibilities.

KDD is defined as a series of procedures to extract knowledge from a dataset (FAYYAD et al., 1996). Figure 1 illustrates its main steps, which are described in the following:

1. **Selection**: identification of the subset of a dataset that is meaningful for the task of analysis. This step is also known as data reduction;

2. **Preprocessing**: data organization and data cleaning, *e.g.*, handling missing data or removing redundant attributes;

3. **Transformation**: it represents the data encoding, *e.g.*, to transform a discrete value into a categorical one. Also, it may include data enrichment that aggregates more information to the instances;

4. **Data Mining**: the most vital step; it refers to the exploration and analysis performed to uncover useful knowledge from the data to be represented by patterns, rules or relationships;

5. **Interpretation**: the last step analyzes the knowledge through interpretation of the patterns discovered. Additionally, an evaluation on these patterns is performed, so that one can return to a previous step whenever the patterns are not considered to be valid, new, potentially useful and ultimately understandable.

![Figure 1 – Knowledge Discovery in Databases](source)
2.1.1 Data Mining

Data Mining (DM) refers to the application of algorithms on data to detect meaningful patterns. It is the main focus of this MSc work. In this procedure, statistical models and Machine Learning (ML) algorithms, i.e., algorithms that learn with experience, are typically used to infer patterns. The decision about what approach should be used commonly depends on characteristics of the data and also on the type of KDD task to be performed. The most common tasks includes: classification, clustering and association rule discovery (Kamber; Han; Pei, 2012; Goldschmidt; Passos; Bezerra, 2015). They are briefly described in the following.

2.1.1.1 Classification

Classification is the process of assigning a class to one record, i.e., to select a value for a categorical attribute among one predefined set of options, based on the values of other attributes and also on example classified instances that are used for training. That is, classification makes a class prediction for one given data instance through data analysis (Kamber; Han; Pei, 2012). The main focus is the search for a model, e.g., a function, that properly describes the data classes. The most common techniques are: k-Nearest Neighbor classifier (Cover; Hart, 1967), Multilayer Perceptron (Werbos, 1974), Bayesian Networks (Pearl, 1988), Genetic Algorithms (Koza, 1992) and C4.5 (Quinlan, 1993).

After finding a classification model, its accuracy needs to be evaluated. In this sense, it is common to measure how good or how accurate the classifier is to predict class labels (Kamber; Han; Pei, 2012). To make it possible, the dataset is usually split into training and testing sets. The training set “teaches” the data behavior for the model, i.e., it is the data used to build the model. Meanwhile, the testing set allows evaluating the model through some metrics, such as accuracy (ACC), precision (PRE), recall (REC) and f-score ($F_1$). For a binary classification, i.e., two classes, these metrics are defined in the following.

\[
\text{ACC} = \frac{TP + TN}{TP + TN + FP + FN}, \text{ range: } [0, 1]
\]
\[
\text{PRE} = \frac{TP}{TP + FP}, \text{ range: } [0, 1]
\]
\[
\text{REC} = \frac{TP}{TP + FN}, \text{ range: } [0, 1]
\]
\[
F_1 = \frac{2 \times \text{PRE} \times \text{REC}}{\text{PRE} + \text{REC}}, \text{ range: } [0, 1]
\]

Let us consider that positive tuples are those that refer to the main class of interest, while negative tuples are all other tuples. Therefore, $TP$, $TN$, $FP$ and $FN$ are:

- True Positives ($TP$): the number of positive tuples correctly labeled;
- True Negatives ($TN$): the number of negative tuples correctly labeled;
• False Positives \((FP)\): the number of negative tuples incorrectly labeled;

• False Negatives \((FN)\): the number of positive tuples incorrectly labeled.

2.1.1.2 Clustering

Clustering divides a dataset into subsets of instances, \(i.e.\), clusters, such that instances of the same cluster share similar characteristics while instances of distinct clusters are dissimilar to each other (XU; WUNSCH, 2009). Among the most traditional clustering algorithms, there are: \(k\)-means (LLOYD, 1982), Fuzzy \(c\)-means (DUNN, 1973) and DBSCAN (ESTER et al., 1996).

When the data has high dimensionality, that is, when there is a numerous set of attributes, traditional clustering algorithms tend to be inefficient and ineffective. To tackle this problem, a number of subspace clustering algorithms have been presented in literature (CORDEIRO et al., 2013; AGRAWAL et al., 1998; MOISE; SANDER, 2008). They search for clusters of instances that only occur when the data is projected into subspaces of the original data space, \(i.e.\), when considering only subsets of the original attributes, so these algorithms automatically spot subspaces with high-density clusters. Extensive search is not a viable solution to this problem since the number of subspaces to analyze grows exponentially with the number of attributes. Thus, it is common the use of approximation strategies and heuristics to make the algorithms more efficient and scalable. Examples of subspace clustering algorithms are: CLIQUE (AGRAWAL et al., 1998), PROCLUS (AGGARWAL et al., 1999), P3C (MOISE; SANDER; ESTER, 2006), STATPC (MOISE; SANDER, 2008) and HALITE (CORDEIRO et al., 2013).

Ground truth labels are unknown for clustering in most cases. Therefore, the accuracy evaluation is commonly performed using the clustering model itself (XU; WUNSCH, 2009). The existing metrics use the clusters and their objects to evaluate the model, assuming that an ideal model minimizes intracluster distances, and still maximizes intercluster distances. Examples of such metrics are: Silhouette (ROUSSEEUW, 1987), Calinski-Harabasz Index (CALINSKI; HARABASZ, 1974), and Davies-Bouldin Index (DAVIES; BOULDIN, 1979).

2.1.1.3 Association Rules

The detection of association rules is the process of identifying sets of data items that frequently occur together in transactions, \(e.g.\), market baskets in e-commerce, so to define rules that represent if-then associations among frequent itemsets. Each rule has one antecedent itemset (if) and one consequent itemset (then), \(i.e.\), if \(\rightarrow\) then; it indicates that whenever a transaction includes the antecedent itemset, it is highly likely that it also has the consequent one (AGRAWAL; IMIELiński; SWAMI, 1993). Algorithms Apriori (AGRAWAL; SRIKANT et al., 1994) and PCY (PARK; CHEN; YU, 1995) are among the most traditional association rule detectors.

The detection of many rules is usually undesirable, mainly because it complicates the result interpretation. In this way, there are some pruning methods based on quality measurements
for rules, such as support \((SUPP)\) and confidence \((CONF)\) (KAMBER; HAN; PEI, 2012). To illustrate this idea, let us consider a dataset \(T\) of transactions and a rule \(X \rightarrow Y\), in which \(X\) is a non-empty antecedent itemset, \(Y\) is a non-empty consequent itemset, and \(X \cap Y = \emptyset\). The support and the confidence of rule \(X \rightarrow Y\) is defined in the following.

\[
SUPP(X) = \frac{\text{number of transactions that include } X}{\text{total number of transactions in } T}, \text{ range: } [0, 1]
\]

\[
CONF(X \rightarrow Y) = \frac{SUPP(X \cup Y)}{SUPP(X)}, \text{ range: } [0, 1]
\]

It is common in Data Mining that the researcher establishes minimum limits for these measurements, to exclude rules that do not fulfill these limits from the result. Therefore, it is important to identify rules with high support and high confidence.

Besides support and confidence, there are other metrics for the dependence between itemsets \(X\) and \(Y\), such as leverage \((LEV)\) (PIATETSKY-SHAPIRO, 1991), lift \((LIFT)\) and conviction \((CONV)\) (BRIN et al., 1997). Leverage measures the difference between the frequency of \(X\) and \(Y\) appearing together and the frequency if \(X\) and \(Y\) were independent, ranging from -1 to 1, where score equal to zero indicates that they are independent. Lift measures the frequency of \(Y\) to be a consequence of \(X\), ranging from 0 to \(\infty\). Finally, conviction measures the probability of \(Y\) be consequent of \(X\) based on the actual frequency of \(X\) without \(Y\), ranging from 0 to \(\infty\). In lift and conviction, score equals to 1 indicates that \(Y\) is independent of \(X\); otherwise, they are dependent.

### 2.2 Player Modeling

The KDD tasks presented so far are applicable in general data domains. However, how to perform knowledge discovery in datasets of games, specifically in players’ data? This is a research area named Player Modeling (PM), whose goal is to extract the knowledge and experience from players based on their actions in games (YANNAKAKIS et al., 2013). Player Modeling is a research area from Artificial Intelligence in Games (Game AI) that focuses on capturing the characteristics and behaviors of players, as responses of the environment in which they interact, in order to model them. PM performs procedures to infer knowledge in games identifying the common patterns and behaviors of the players (YANNAKAKIS, 2012; YANNAKAKIS et al., 2013; YANNAKAKIS; TOGELIUS, 2018). For instance, one can model influencers to detect them.

PM is commonly split into two approach types: model-based and model-free techniques. The model-based ones, also known as top-down approaches, are techniques based on Social & Human Sciences hypotheses that explore phenomena of emotion, sentiment and sensations of the player. They are often induced by empirical observation experiments (YANNAKAKIS et al.,
Meanwhile, the model-free techniques, also known as bottom-up approaches, focus on the creation of a computational representation of the player knowledge and his/her state in a particular game. These techniques assume that the model can be developed based on data collected from games. Clustering, classification and statistical analyses are commonly used to develop the model; the best strategy depends on the required knowledge in most cases (YANNAKAKIS et al., 2013; YANNAKAKIS; TOGELIUS, 2018).

PM has many applications; they vary from the comprehension of players’ experiences for direct marketing to game design personalization based on preferences of players (YANNAKAKIS; TOGELIUS, 2015). For example, it is possible to model digital influencer players in order to detect them, by transforming this problem into a classification task. In fact, it is one of the main goals of this MSc work; see Chapter 4. In order to increase the attractiveness of the game, it is also possible to develop a game design based on the knowledge of players’ preferences. To extract such knowledge is the second main goal of this MSc work; see Chapter 5. Anyway, we focus on identifying these characteristics on Social Networks of Games. In this sense, the next section explores background concepts from social networks.

2.3 Social Networks

In a Social Network, a mathematical structure called graph (or network) represents the objects and their relationships computationally. This structure is used to extract or infer knowledge from a relationship network (GOLDSCHMIDT; PASSOS; BEZERRA, 2015; BARABÁSI; PÓSFAI, 2016). This section presents the fundamental concepts and definitions on this subject.

2.3.1 Network Terminology

A graph is a structure formed by three parameters (BARABÁSI; PÓSFAI, 2016): a set \(V\) of vertices, \(i.e.,\) objects of the network, also known as nodes; a set \(E\) of edges, also known as connections or links; and a function \(\delta\) of \(E\) in \(V \times V\) that relates pairs of vertices to edges. The notation used to refer to a graph is \(G(V,E,\delta)\), or simply \(G(V,E)\).

In a graph \(G(V,E)\), the links can be directed or undirected. If it is not informed, the graph is considered to be undirected. One undirected connection means that an edge \(e \in E\) is associated with a set of vertices \(\{a,b\} \subseteq V\), that is, \(\delta(e) = \{a,b\}\), representing a mutual relationship, \(e.g.,\) player \(a\) is friend of player \(b\), and the opposite is also true. Meanwhile, a directed graph has directed links. A directed link means that an edge \(e \in E\) is associated with one ordered pair of vertices \((a,b)\), where \(a,b \in V\), that is, \(\delta(e) = (a,b)\), representing a direct relationship, \(e.g.,\) a player \(a\) killed a monster \(b\), or a monster \(b\) was killed by a player \(a\).

Figure 2 shows one example of a directed graph \(G(V,E)\) with six nodes connected by five edges. In this network, \(V = \{v_1,v_2,v_3,v_4,v_5,v_6\}\) and \(E = \{e_1,e_2,e_3,e_4,e_5\}\), where \(\delta = \{(v_1,v_3),(v_2,v_3),(v_3,v_4),(v_4,v_5),(v_6,v_4)\}\).
Figure 2 – Example of a directed graph

Another characteristic of a graph is the type of its edges. There are two types: weighted and unweighted edges. Unweighted edges are connections without an explicit value, which means that all edges in the graph have the same weight, i.e., the same importance. On the other hand, weighted edges have explicit values, in which each edge $e_i \in E$ has a value $w_i$ associated to it to represent the weight of the respective link. Thus, in a graph with weighted edges, it is possible to represent strong and weak connections quantitatively. For example, consider a player-game network where players evaluate games using numbers of stars, so that the more stars a player gives to a game, the better is the game in the player’s opinion.

Additionally, there are types of graphs, such as complete graphs, bipartite graphs, connected graphs and subgraphs. A complete graph is a network in which each node is connected to all other vertices. The notation used for a complete graph is $K_n$, in which $n$ is the number of nodes. Figure 3a illustrates a toy complete graph $K_4$.

**Definition.** A complete graph $K_n(V,E)$ is a graph where each node $v_i \in V$ has a direct connection to all other nodes, i.e., $\forall v_j \in V \ | \ v_j \neq v_i$. If it is a directed graph, ordered pairs $(v_i, v_j)$ are used; otherwise, $\{v_i, v_j\}$. Equation 2.1 provides the formal definition:

$$\forall v_i, v_j \in V \ [\exists (v_i, v_j) \lor \{v_i, v_j\} \in \delta] \ | \ v_i \neq v_j$$  \hspace{1cm} (2.1)

A graph can also be bipartite (or bigraph). The set of nodes $V$ of a bigraph $G_b(V,E)$ can be split into two disjoint sets $V_x$ and $V_y$, in which vertices of $V_x$ are only connected to vertices
Chapter 2. Fundamental Concepts

of $V_y$, and the opposite is also true. For instance, a toy bigraph is shown in Figure 3b. It has
$V_x = \{v_1, v_2, v_5\}$ and $V_y = \{v_3, v_4, v_6\}$.

**Definition.** A bipartite graph $G_b(V, E)$ comprises a graph with nodes $V = V_x \cup V_y \mid V_x \cap V_y = \emptyset$ and a set of edges $E \subseteq V_x \times V_y$.

In addition, a graph can be connected or not. A connected graph $G_c(V, E)$ is a network in which there is a bridge between all possible pairs of distinct nodes, that is, they are directly or indirectly connected to each other. A node $v_i \in V$ is directly connected to a node $v_j \in V$ iff $v_i$ and $v_j$ have a link in $\delta$. Therefore, in an undirected graph, the undirected connection $\{v_i, v_j\} \subset \delta$ must exist; in a directed graph, the corresponding exigence is $(v_i, v_j) \in \delta$. Also, a node $v_i \in V$ is indirectly connected to another node $v_k \in V$ iff there is a path of nodes directly connected to each other that starts in $v_i$ and ends in $v_k$. For example, in an undirected graph, a vertice $v_i$ is indirectly connected to other vertice $v_k$ when there is a path of directly connected nodes between them, i.e., $\{v_i, v_j\}, \{v_j, v_{j+1}\}, \ldots, \{v_{k-1}, v_k\} \subset \delta$.

**Definition.** Exist a bridge between $v_i$ and $v_j$ when there is a path of directly or indirectly connected to each other. Formally defined by:

$$v_i \Rightarrow v_j$$

**Definition.** A connected graph $G_c(V, E)$ comprises a network in which all possible pairs of distinct nodes, i.e., $\forall \{v_i, v_j\} \subset V \mid v_i \neq v_j$, present a bridge $v_i \Rightarrow v_j$. It is formally defined in Equation 2.2:

$$\forall \{v_i, v_j\} \subset V \left( \exists v_i \Rightarrow v_j \mid v_i \neq v_j \right)$$

On the other hand, a disconnected graph is a network in which there is at least one pair of disconnected nodes. For example, the toy graph in Figure 3c is disconnected, since it has two disjoint sets of vertices $V_x = \{v_1, v_2, v_3\}$ and $V_y = \{v_4, v_5, v_6\}$, such that for any vertices $v_i \in V_x$ and $v_j \in V_y$, a bridge $v_i \Rightarrow v_j$ is missing, so $v_i \not\Rightarrow v_j$. Note, the sets $G_x$ and $G_y$ are part of the graph $G_d$, which by definition are subgraph of the graph $G_d$, so $G_d \supseteq G_x \cup G_y$.

**Definition.** A subgraph $G_x$ is part of a graph $G$, in which the nodes and links of $G_x$ are also in $G$. It is formally defined in Equation 2.3:

$$G_x(V_x, E_x) \subseteq G(V, E) \mid V_x \subseteq V, E_x \subseteq E, \delta_x \subseteq \delta$$

### 2.3.2 Network Metrics

How to quantify properties of a graph? Graphs have several types of measurements. The primary ones are based on counting. Given a graph $G(V, E)$, the count of nodes is represented
by \( n = |V| \), i.e., \( n \) means the number of vertices of the network. Similarly, the count of edges is represented by \( m = |E| \). For example, Figure 4 illustrates a network \( G \) with \( n = 6 \) and \( m = 7 \).

Figure 4 – Example of an undirected graph

![Figure 4](source: Elaborated by the author.)

One of the most important metrics in a graph is the degree (BARABÁSI; PÓSFAI, 2016). It represents the number of links of a node. Considering that the degree of a node \( v \) is denoted as \( g(v) \), the degree of nodes in Figure 4 are: \( g(v_1) = 2 \), \( g(v_2) = 2 \), \( g(v_3) = 3 \), \( g(v_4) = 3 \), \( g(v_3) = 2 \) and \( g(v_6) = 2 \). In weighted graphs, the number of edges of a node \( v \) is \( m_v \), while the degree \( g(v) \) of \( v \) is the sum of weights \( w_i \) of all of its edges, as shown in Equation 2.4.

\[
g(v) = \sum_{i=1}^{m_v} w_i
\]  

(2.4)

Given the capacity of directed graphs to represent the direction of each link, there are distinct degrees for input and output connections of the vertices. The input degree \( g_{in}(v) \) represents the count of edges that arrive on node \( v \), while the output degree \( g_{out}(v) \) is the count of edges that start from node \( v \). For instance, in Figure 2, the degrees of the nodes are: \( g_{in}(v_1) = 0 \), \( g_{out}(v_1) = 1 \), \( g_{in}(v_2) = 0 \), \( g_{out}(v_2) = 1 \), \( g_{in}(v_3) = 2 \), \( g_{out}(v_3) = 1 \), \( g_{in}(v_4) = 2 \), \( g_{out}(v_4) = 1 \), \( g_{in}(v_5) = 1 \), \( g_{out}(v_5) = 0 \), \( g_{in}(v_6) = 0 \) and \( g_{out}(v_6) = 1 \).

In addition, there is the density \( d(G) \) of a graph \( G \); it is a global metric on the edges. The density of a graph \( G(V, E) \) with \( n = |V| \) nodes and \( m = |E| \) edges quantifies how close are \( m \) and the number of edges of a complete graph \( K_n \). In this way, a high density means that \( G \) is close to the complete graph; otherwise, a low density means that \( G \) is close to a graph with no edges. When \( G \) is a directed graph, its density is given by Equation 2.5. As can be seen, the density of a directed graph can be computed by considering input or output degrees.

\[
d(G) = \frac{m}{n(n-1)} = \frac{\sum_{v \in V} g_{in}(v)}{n(n-1)} = \frac{\sum_{v \in V} g_{out}(v)}{n(n-1)}
\]  

(2.5)

When the graph is undirected, the links are mutual, so it is necessary to double the number of edges to calculate the density. For one undirected graph \( G(V, E) \), its density \( d(G) \) is given by \( 2m \) divided by the maximum number of edges \( n(n-1) \), as defined in Equation 2.6.

\[
d(G) = \frac{2m}{n(n-1)} = \frac{\sum_{v \in V} g(v)}{n(n-1)}
\]  

(2.6)
Also, there is the sparsity \( s(G) \) of a graph \( G \), which is the opposite of density. In this sense, a high sparsity \( s(G) \) means that \( G \) is close to a graph with no edges; low sparsity means that \( G \) is close to \( K_n \). Formally, its is defined in Equation 2.7.

\[
s(G) = 1 - d(G)
\]  

(2.7)

### 2.3.3 Social Network Analysis

One graph can model a social network, a map of cities, a network of computers, player-game interactions and many other kinds of relationship-based data (BARABÁSI; PÓSFAI, 2016). Similarly to the general KDD, the task known as Social Network Analysis (SNA) seeks to extract knowledge from relationship-based data of social networks using as a basis the mathematical structure of graphs. SNA is commonly split into three tasks (GOLDSCHMIDT; PASSOS; BEZERRA, 2015; BARABÁSI; PÓSFAI, 2016): (1) community detection, e.g., to infer communities of similar players; (2) link prediction, e.g., to infer friendship links among players, and; (3) ranking, e.g., to sort players by their ability to influence other players.

As aforementioned, one of the main goals of this MSc work is to identify digital influencers in games, i.e., game influencers. This problem can be understood as a ranking task, so we describe in the following well-known ranking strategies that are available in the literature. Generally, the ranking task comprises the analysis of a graph to sort its nodes or edges according to a predefined criterion, like importance and popularity (GOLDSCHMIDT; PASSOS; BEZERRA, 2015).

The degree centrality is the most primary ranking method (BARABÁSI; PÓSFAI, 2016). It refers to sorting the vertices by their degrees. The degree centrality is an attractive method because it is agnostic about the content of the vertices, and it can also consider only specific types of relationships of interest, e.g., to play, to like or to share a game. Other well-known ranking approaches are: closeness centrality, betweenness centrality and more sophisticated algorithms, such as the PageRank (PAGE et al., 1999). The closeness centrality measures how much a vertice of interest is close to all other vertices (GOLDSCHMIDT; PASSOS; BEZERRA, 2015). Differently, the betweenness centrality quantifies how much a vertice of interest assumes the role of bridge in the shortest path between pairs of other vertices (GOLDSCHMIDT; PASSOS; BEZERRA, 2015; BARABÁSI; PÓSFAI, 2016). PageRank was designed to sort web pages by their importance; it is a vital component of the Google’s search engine1. PageRank implements the recursive concept that a web page, i.e., a vertice, is important if other important pages point to it; that is, if it has many in-links from pages with many in-links (PAGE et al., 1999). Finally, it is worth noting that there are many other ranking methods in the literature (BARABÁSI; PÓSFAI, 2016). In most cases, the decision about what approach should be used depends on the problem.

---

1 Google. <http://google.com/> (accessed July 08, 2019).
2.4 Discussion

Player Modeling extracts or infer the experience of players when using games with the aim of supporting the development of new games that are more attractive. It commonly uses KDD techniques to identify patterns and rules on players’ data. Therefore, we discussed in this chapter the main KDD concepts, as well as Social Network concepts. Social Network Analysis searches for knowledge on relationship-based data represented as graphs. As discussed before, data from several applications can be modeled as Social Networks, e.g., interactions between players and games, and processed using Social Network Analysis. The next chapters describe the main contributions of this MSc work exploring Social Network and KDD techniques. Due to the diversity of concepts involved, each chapter has specific sections for related work, methodology and experimental evaluation.
CRAWLING A SOCIAL NETWORK OF GAMES

With the expanding games’ universe, the application of knowledge discovery is growing due to the need to make games more attractive and intelligent (YANNAKAKIS; TOGELIUS, 2015). Researchers focus on understanding the players’ behaviors, *i.e.*, Player Modeling (YANNAKAKIS et al., 2013; YANNAKAKIS; TOGELIUS, 2018). From the perspective of game designers, players’ behavior is one of the most important factors that must be considered when developing new game systems (LEE et al., 2011; YANNAKAKIS; TOGELIUS, 2018).

An ongoing research topic is PM in platforms of video games. A platform of video games is a virtual environment in which players interact with games (JIA et al., 2015; EBERHARD et al., 2018). The player-game interactions represent relationships of several types, *e.g.*, to buy, to play and to comment on a game. Each relationship can be seen as a social network; in this context, we coin the term Social Network of Games (SNG) for them.

This chapter presents the new dataset **SMMnet**; crawled from a real SNG, it is one of the three main contributions of this MSc work. At first, Section 3.1 discusses the related work. Then, the concept of Social Network of Games is introduced in Section 3.2. A real platform of video games that can be understood and analyzed as a SNG is then presented in Section 3.3. Section 3.4 describes the data collection and the schema of **SMMnet**, for which it also reports statistics. Finally, the applicability of our dataset is discussed in Section 3.5, while the last Section 3.6 concludes the chapter.

### 3.1 Related Work

Players produce many types of data during a gameplay. There are four main types: (1) player data, *e.g.*, scores, items collected and number of lives; (2) avatar state, *e.g.*, jumping, crouching and running; (3) game state, *e.g.*, level of current stage and number of monsters;
and (4) telemetry data, e.g., event logs that refer to moment and action records. In addition, player-game interactions are complex and can produce a vast amount of data (YANNAKAKIS et al., 2013; YANNAKAKIS; TOGELIUS, 2015).

Nowadays, there are several game datasets publicly available. Some of them focus on commercial games (LEE et al., 2011; JOHNSON et al., 2016; LIN et al., 2017; AUNG et al., 2018); others refer to independent games (LIM; HARRELL, 2015; KARPOUZIS et al., 2015). However, there is a scarce exchange of knowledge between academic and industrial research in games (YANNAKAKIS, 2012). Game companies do not make their games and datasets available because they can expose crucial information about their products. For instance, revealing the game engine and players’ statistics may allow bug identification, or it may pave the way for malicious users to develop cheating systems (ZOELLER, 2013).

An industrial initiative came from Microsoft; named Malmo (JOHNSON et al., 2016). Malmo is a system to support artificial intelligence experimentation on the game Minecraft (Microsoft, Washington, United States). It performs tasks related to monitoring, routing and collaborative contribution. Minecraft is a game in which the player is exposed to a 3D digital world with many challenges. Thereby, Malmo is a development engine; not a dataset. However, it can be used to elaborate one.

To the best of our knowledge, MOBA (AUNG et al., 2018) is the only dataset studied in the literature that came from an industrial initiative. It is based on the famous game League of Legends (Riot Games, California, United States). Provided by the original game developers, it contains one year of telemetry data collected from 400 thousand players. Interestingly, MOBA was used by researchers to explore relationships that exist between behavior and cognitive psychology of game players (AUNG et al., 2018).

There are also public datasets developed by independent researchers, i.e., without the support of the game companies. Some public gaming datasets are: World of Warcraft Avatar History Dataset (WoWAH) (LEE et al., 2011), Platformer Experience Dataset (PED) (KARPOUZIS et al., 2015) and StarCraft AI Research Dataset (STARDATA) (LIN et al., 2017).

WoWAH is a dataset based on the massively multiplayer online game World of Warcraft (Blizzard Entertainment, California, United States). It contains telemetry data from over 91 thousand avatars collected between January 2006 and January 2009. This dataset was elaborated independently by Lee et al. (2011) through a computer script that collected a list of all online avatars at every 10 minutes during the three years of the study.

PED is the first game experience corpus that contains multiple modalities of user data from players of the game PlatformerAI, which is a prototype game based on the well-known, commercial game Super Mario Bros (Nintendo, Kyoto, Japan). This dataset supports the analysis of players’ behaviors by including player data, game state data and gameplay recordings (KARPOUZIS et al., 2015).
3.2 Social Network of Games

STARDATA is a dataset with 65,646 replays from the game StarCraft (Blizzard Entertainment, California, United States). It contains 1,535 million frames and 496 million players’ actions. Moreover, the game state was registered continuously with an interval of three frames. The replays were collected manually from four public websites of StarCraft fans (LIN et al., 2017).

Table 1 summarizes the related work and our proposal. Each of the existing datasets has a specific purpose. Nevertheless, they are constrained in PM and game content exploration. To the best of our knowledge, none of theses datasets available include relationship-based data collected from a Social Network of Games, thus making it difficult to perform Social Network Analysis in this context, like community detection, link prediction and ranking. This MSc work shrinks the problem with the proposal of SMMnet: the first SNG dataset.

| Name                    | Public | SNG |
|-------------------------|--------|-----|
| WoWAH (LEE et al., 2011)| Yes    | Not |
| PED (KARPOUZIS et al., 2015) | Yes    | Not |
| STARDATA (LIN et al., 2017) | Yes    | Not |
| MOBA (AUNG et al., 2018)  | Not    | Not |
| SMMnet (ours)           | Yes    | Yes |

Source: Elaborated by the author.

3.2 Social Network of Games

As discussed in Section 2.3, a Social Network describes interactions and relationships of users in an environment (BARABÁSI; PÓSFAI, 2016). In a network, user relationships are represented by links. There are many types of links, e.g., links denoting social connections, similarity, behavioral interactions, actions and the like (SAVIĆ; IVANOVIĆ; JAIN, 2019). Links may also be present in a platform of video games, where players perform many types of interactions with games (JIA et al., 2015; EBERHARD et al., 2018). In these platforms, a player can buy or play a game, comment on a game and perform a number of other interactions. Figure 5 illustrates two players $p_1$ and $p_2$ interacting, i.e., buying, playing and commenting on, with three games $c_1$, $c_2$ and $c_3$. Each of these types of relationship forms a social network; therefore, we coin the term Social Network of Games (SNG) to refer to them in this MSc work.

In this sense, each relationship forms a social network represented by a directed bipartite graph $G = (V, E)$ with edges $e \in E$ of players $p \in V$ interacting with games $c \in V$, so $e = (p, c)$. These networks change over time because new players and games arrive, and new links appear. Therefore, since each graph changes as a function of time, it is a dynamic network (WESTABY, 2012). Thereby, there are a series of snapshots of static graphs over time.
Note that the interactions in a platform of video games are represented by a set of networks. Therefore, we can study each relationship network individually, as independent SNGs. This is the default configuration used in the MSc work; it was chosen due to the fact that many algorithms of Social Network Analysis work only on homogeneous networks. Note, however, that the same interactions can also be represented as a single complex network, i.e., a single SNG with many types of interactions all together, by allowing the vertices to be connected by edges of different and possibly overlapping types (CHERIFI et al., 2017). In this way, one can take advantage of existing analytical algorithms in the field of complex networks to extract knowledge from player-game interactions. Also, each node can be represented as a complex object; for example, by storing game features, like price, type of game and developer, as well as player features, such as age, gender and average daily time of playing.

### 3.3 Super Mario Maker

Super Mario Maker (SMM) is a platform of video games developed by Nintendo (Kyoto, Japan) for the Nintendo Wii U\(^1\) and Nintendo 3DS/2DS\(^2\) consoles. It was launched in September 2015. In this platform, a player can like, play and comment on game levels based on the Super Mario Bros series. Also, the player can create new game levels and share them online with the world. In this sense, SMM can be represented as a Social Network of Games.

#### 3.3.1 Platform

In SMM, players make several types of relationships; they are illustrated in Figure 6. A player can (1) create a game level and (2) play levels created by other players. If a player completes the challenge of the game level, he/she (3) “cleared” the level. The player can also be the (4) first person to clear or beat the (5) time record of a level. Finally, the player can (6) like any game level played. In this sense, this Social Network of Games has six types of relationships.

---

\(^1\) Nintendo Wii U. \(<http://bit.ly/nintendo-wii-u-smm>\) (accessed July 02, 2019).

\(^2\) Nintendo 3DS/2DS. \(<http://bit.ly/nintendo-3ds-smm>\) (accessed July 02, 2019).
3.3. Super Mario Maker

Players can develop game levels based on the Super Mario Bros series, besides playing predefined levels or levels developed by other players. The game levels can be developed in four styles. As illustrated in Figure 7, the styles available are: Super Mario Bros, Super Mario Bros 3, Super Mario World and Super Mario Bros U. Any new game level must be “cleared” by its creator, i.e., maker, to be published online.

Figure 7 – Game styles in Super Mario Maker

![Game styles](image)

(a) Super Mario Bros  
(b) Super Mario Bros 3  
(c) Super Mario World  
(d) Super Mario Bros U

Source – Nintendo. <http://supermariomaker.nintendo.com/> (accessed July 16, 2019).

3.3.2 SMM Bookmark

The website SMM Bookmark\(^3\) reports metadata about Super Mario Maker publicly. Figure 8 shows a snapshot of an online game level’s page, in which allows to access some game information, such as game difficulty, name, style, creation date, stars, i.e., number of likes, photos of the level and its maker. For each game level, one can also access the latest player-game

\(^3\) SMM Bookmark. <https://supermariomakerbookmark.nintendo.net/> (accessed July 16, 2019).
interactions, as shown in Figure 9. Note that allows to identify the latest player that beat the time record, besides the one that made the first clear, and who performed the latest plays, latest clears and latest likes. Additionally, SMM Bookmark makes available a website for each player with detailed activities. Figure 10 reports a player website, in which allows to access his/her name, nationality, number of stars received by levels developed, number of games played, number of games cleared, among others.

Figure 8 – SMM Bookmark: Game level

![Normal](https://supermariomakerbookmark.nintendo.net/)

Source – SMM Bookmark. <https://supermariomakerbookmark.nintendo.net/> (accessed July 16, 2019).

Figure 9 – SMM Bookmark: Player-game interactions

![World Record](https://supermariomakerbookmark.nintendo.net/)

Source – SMM Bookmark. <https://supermariomakerbookmark.nintendo.net/> (accessed July 16, 2019).

3.4 SMMnet Dataset

In this section, we present the Super Mario Maker Network Dataset (SMMnet). To the best of our knowledge, it is the first very large (as of May, 2019, when it was publicly released) and open access dataset of a Social Network of Games. SMMnet provides information about over
3.4. **SMMnet Dataset**

Figure 10 – SMM Bookmark: Player’s statistics

| Stars Received | Medals Earned |
|----------------|--------------|
| ★115           |              |

| 100 Mario Challenge | Play History |
|---------------------|--------------|
| Easy clears         | Courses played | 3362 |
| Normal clears       | Courses cleared | 2151 |
| Expert clears       | Total plays    | 9555 |
| Super Expert clears | Lives lost     | 7394 |

Source – SMM Bookmark. <https://supermariomakerbookmark.nintendo.net/> (accessed July 16, 2019).

115 thousand game levels and over 880 thousand players that performed nearly seven million interactions of different types on the levels. We describe in the following the data collection process and the schema of SMMnet for a Relational Database Management System (RDBMS); then, we report statistics about the data.

### 3.4.1 Data Collection

Metadata from SMM is publicly reported on SMM Bookmark. Thus, we elaborated a web crawler to collect it. According to Areekijseree, Laishram and Soundarajan (2018), a web crawler (or simply crawler) is a computer program that navigates on the World Wide Web in a systematic and automated way to search for or to collect data. In order to build our crawler, we searched for any available Application Programming Interface (API) that could retrieve information from the bookmark; and, we found a set of APIs developed in Node.js programming language by independent programmers, *i.e.*, outside Nintendo. However, the APIs available do not collect players’ data nor they keep track of new games developed and player-game interactions overtime, like new plays, records and clears. Hence, we needed to adapt the most recent API, *i.e.*, `super-mario-maker-client`, to keep track of new interactions. Also, we had to develop two novel APIs: (i) `smm-maker-profile` to collect players’ data, and; (ii) `smm-course-search` to search for new game levels developed recently. Both APIs are publicly available for download⁴.

Figure 11 illustrates our crawler. It runs constantly with API `smm-course-search` searching for new game levels and storing their identifiers into a database. The identifiers are then read by API `super-mario-maker-client` that collects the corresponding levels’ data from SMM

---

⁴ npm. <https://www.npmjs.com/~leomaurodesenv> (accessed June 29, 2019).
Bookmark, while API *smm-maker-profile* collects data from any player that interacted with the levels. In our current implementation, the database system PostgreSQL\(^5\) supports the data storage; it was chosen since PostgreSQL is a robust, open-source and popular system. Finally, it is important to note that our crawler respects the access policies of the website SMM Bookmark, *i.e.*, those in file “robots.txt”.

![Figure 11 – Crawler: Data collection structure](image)

Source: Elaborated by the author.

However, the data collector suffered with network bandwidth limitations, as the servers of SMM Bookmark responded to a page request in around 2.7 seconds. Thus, it was impracticable to explore the whole network. Due to this lack of scalability, we had to focus on a relatively small dataset sample with information related to nearly 115,000 game levels. The sample was carefully chosen to maximize the edge coverage over clusters of nodes, so to preserve the community structure of the network, as suggested by Areekijserre, Laishram and Soundarajan (2018).

Therefore, we collected data from 4 nationalities: France (FR), Germany (DE), Canada (CA) and Brazil (BR). We selected FR, DE and CA for being the most active communities, right after United States (US) and Japan (JP), for which too many levels were created per day, thus making it impractical to keep track of. We also selected BR to represent South America. The data collection was performed during almost five months, from 16 November 2017 to 10 April 2018. A new snapshot of the network of these countries’ game levels was collected at every two hours.

### 3.4.2 Data Description

Dataset SMMnet is stored into seven files in the Comma-Separated Values (CSV) data format: (1) COURSES.CSV with static game levels’ data; (2) COURSE-META.CSV with dynamic data on game levels; (3) PLAYERS.CSV with static players’ data; (4) PLAYS.CSV with play interactions; (5) CLEARS.CSV with clear interactions; (6) LIKES.CSV with like interactions, and;

\(^5\) PostgreSQL. [https://www.postgresql.org/](https://www.postgresql.org/) (accessed July 16, 2019).
3.4. SMMnet Dataset

(7) RECORDS.CSV with time record interactions. Figure 12 shows a schema with non-normalized relations that can be used to store SMMnet into a RDBMS; it has seven tables, one per CSV file.

Figure 12 – Relational schema for SMMnet

Source: Elaborated by the author.

Relation COURSES stores static metadata about SMM levels. It includes the identifier of each level, besides a foreign key referring to the player who created the level, the timestamp of creation and other static information on the level. The corresponding temporal-related metadata is stored in relation COURSE-META. It includes counts of players, stars, tweets, clears and attempts per level, besides a tag, the clear rate, a foreign key referring to the player that first cleared the level and the timestamp of the data collection. Relation PLAYERS stores static information on the players, including their identifiers, avatar images, flags and names. Finally, relations PLAYS, CLEARS, LIKES and RECORDS store player-game interactions, i.e., links of the SNG, through many-to-many relationships between relations COURSES and PLAYERS with the timestamp of the data collection. The world time records in milliseconds are also stored in relation RECORDS.

Table 2 reports the cardinality of each relation. Dataset SMMnet includes 115 thousand levels that were played 3.94 million times, cleared 2.05 million times with 117 thousand time records and liked 619 thousand times by 884 thousand players around the world. The data is split into seven files that can be easily stored into any RDBMS, such as PostgreSQL, Oracle and MySQL. The next section characterizes SMMnet by reporting statistics about it.

---

6 PostgreSQL. <https://www.postgresql.org/> (accessed July 16, 2019).
7 Oracle. <https://www.oracle.com/br/database/> (accessed November 01, 2019).
8 MySQL. <https://www.mysql.com/> (accessed November 01, 2019).
Table 2 – Relation cardinality in SMMnet

| Relation     | Cardinality |
|--------------|-------------|
| COURSES      | 115k        |
| COURSE-META  | 292k        |
| PLAYERS      | 884k        |
| PLAYS        | 3,940k      |
| CLEARS       | 2,050k      |
| LIKES        | 619k        |
| RECORDS      | 117k        |

Source: Research data.

3.4.3 Data Characterization

Figure 13 and Table 3 report the proportions of levels per game style. As can be seen, 51.7% of the levels follow the game style Super Mario Bros U, 21.2% follow Super Mario Bros, 19.0% follow Super Mario World and 8.1% follow Super Mario Bros 3. Note that there is a high preference for the most recent game style Super Mario Bros U and for the oldest game style Super Mario Bros. In this sense, we presume that the most recent style prevails through the new visual and modern gameplay characteristics, while the oldest style stands out due to the memory of the most traditional games in the Super Mario Bros series.

Table 3 – Game level styles

| Style      | Number of levels |
|------------|------------------|
| marioBrosU | 59.5k (51.7%)    |
| marioBros  | 24.3k (21.2%)    |
| marioWorld | 21.9k (19.0%)    |
| marioBros3 | 9.3k (08.1%)     |

Source: Research data.

Table 4 – Game level difficulties

| Difficulty | Number of levels |
|------------|------------------|
| easy       | 29.7k (25.8%)    |
| normal     | 51.4k (44.6%)    |
| expert     | 27.8k (24.1%)    |
| superExpert| 6.2k (05.4%)     |

Source: Research data.
Similarly, Figure 14 and Table 4 report the proportions of game levels according to how difficult or challenging they are. In order of difficulty, 25.8% of the levels are considered to be easy, 44.6% are normal, 24.1% are in the expert degree of difficulty and 5.4% are in the super expert degree. To the best of our knowledge, there is no rule publicly reported to define the difficulty of a game level; SMM platform establishes it automatically.

The total number of interactions, i.e., plays, clears, records and likes, made by players is 6,729,000. Among them, 3,941,378 refer to interactions of the type play, while 2,051,809 are clears, 618,687 are likes and 117,126 are time records. Figure 15 reports the types of interactions performed on the top-100 levels with the highest numbers of interactions. As can be seen, many plays, likes and clears exist, but very few time records occur because the occurrence probability of one new time record for a level tends to decrease fast as the number of past records increase.

Figure 15 – The rank of game levels by number of player’ interactions for the top-100 most popular levels

Figure 16 – The probability density function for the top-100 most popular game levels with power-law and log-normal estimates

Figure 17 – The cumulative distribution function for the top-100 most popular levels with power-law and log-normal estimates

Figure 18 – The complementary cumulative distribution function for the top-100 most popular game levels with power-law and log-normal estimates

The rank in Figure 15 shows there is a fast decrease in the numbers of interactions with very few game levels having many interactions. This behavior is similar to a power-law; a very common pattern in Social Networks that indicates there exist few popular instances and many unpopular ones (NEWMAN, 2005). In the whole dataset, only 0.13% of the levels (151 levels) have more than 500 interactions, while the vast majority of the levels, i.e., 99.87%
(114,881 levels), has interaction counts ranging from 0 to 500. Figure 16 reports a probability density function plot for the top-100 most popular game levels. As can be seen, it exhibits a similar behavior of a power-law (CLAUSEN; SHALIZI; NEWMAN, 2009) and a log-normal distribution estimates (MITZENMACHER, 2004). Also, the same occurs in the cumulative distribution function plot and in the complementary cumulative distribution function plot, as shown in Figures 17 and 18, respectively.

Overall, dataset SMMnet describes nearly seven million interactions of distinct types that were performed by over 880 thousand players on over 115 thousand game levels. As aforementioned, the data refers to four distinct nationalities. Figure 19 and Table 5 summarize this information; they show that ~41 thousand levels were created in FR, ~37 thousand are from DE, ~34 thousand are from CA and ~3 thousand are from BR. Therefore, we argue that dataset SMMnet has a substantial amount of data to make it viable to infer knowledge about players and games in at least four different countries. Additionally, there are no missing values.

Figure 19 – Game levels by country

| Country      | Number of levels |
|--------------|-----------------|
| France (FR)  | 40.5k (35.2%)   |
| Germany (DE) | 37.3k (32.4%)   |
| Canada (CA)  | 34.5k (30.0%)   |
| Brazil (BR)  | 2.7k (02.4%)    |

Source: Research data.

3.5 Applicability

Our SMMnet may be useful for several applications, including studies of Social Networks and Artificial Intelligence in Games. This section briefly describes potential applications aimed at motivating researchers to use our dataset and also to find other creative forms of use for it.

3.5.1 Social Network Analysis

As discussed before, dataset SMMnet can be interpreted as static graphs, dynamic graphs or complex networks. Therefore, many of the existing analytical algorithms from the fields of social networks and complex networks can be used to study it. In this sense, we emphasize that our dataset can be used in different scenarios, by considering each one of the aforementioned setups. It can be explored in social network studies, including, but not limited to: (1) community
3.6. Discussion

detection, e.g., to identify communities of similar players or similar game levels to support targeted marketing; (2) link prediction, e.g., to infer what game levels a player will probably like or play in the future, and; (3) ranking, e.g., to sort games by popularity or players by their capacity to influence others.

3.5.2 Artificial Intelligence in Games

Game companies use artificial intelligence techniques to produce new games with better graphic and interactive qualities (LUCAS, 2009; YANNAKAKIS, 2012). Games are also advantageous test-beds to algorithms because they offer a simulated environment in which many factors react simultaneously (METOYER et al., 2010; YANNAKAKIS, 2012). In this scenario, our new dataset SMMnet has a number of applications. We highlight some possibilities in the following: (1) supervised data mining, e.g., to identify digital influencers among the players by means of classification; (2) unsupervised data mining, e.g., to cluster popular game levels or to uncover association rules about their main characteristics aimed at identifying what makes them attractive, and; (3) player modeling, e.g., to extract characteristics from the players’ activities in the social network aimed at understanding their preferences. We highlight that SMMnet has already been used in the first two of the aforementioned studies, which corroborates its usefulness. Latter in Chapters 4 and 5 we show how it helped us to automatically spot game influencers among thousands of players and to identify the characteristics that may make some game levels more popular than others.

3.6 Discussion

In this chapter, we presented dataset SMMnet. To the best of our knowledge, it is the first SNG dataset. SMMnet represents a set of social networks in which players perform many types of interactions with game levels, e.g., to build, to play and to like a game level, which can also be seen as a single complex network. Additionally, these networks change over time; therefore, they are in fact dynamic graphs as a function of time.

SMMnet was extracted from the well-known platform of video games Super Mario Maker (Nintendo, Kyoto, Japan). It presents a collection of over 880 thousand players that performed nearly seven million interactions of distinct types on over 115 thousand game levels. Additionally, we presented a schema to store the data into a Relational Database Management System, and reported some statistics about it. Finally, we highlighted the applicability of SMMnet in the research fields of Social Networks and Artificial Intelligence in Games. In addition to the example applications discussed in this chapter, we believe that researchers and game designers will find further creative proposals for our dataset. However, it is worth noting that SMMnet does not include internal data of the game levels, like number, type and position of opponents or interactive objects. Only metadata from the social networks of player-game interactions is
available. In the next chapter, we describe a technique developed to detect the most influential users in a SNG; it is the second of the three main contributions of this MSc work.
Digital influencers are users with high influence in the creation of new trends. They exist since the popularization of social media (PEI; MORONE; MAKSE, 2018), and produce digital contents like online videos, posts or forums that attract the attention of many other users. Among them, there are the game influencers that produce digital contents related to digital games (GROS et al., 2018; HILVERT-BRUCE et al., 2018). Players follow the game influencers looking for entertainment and credible information about the universe of games (SJöBLOM; HAMARI, 2017; GROS et al., 2018). In the rest of this monograph, we use the terms game influencer and influencer interchangeably.

**DEFINITION.** A game influencer, or just influencer, is a game level developer that attracts other players using social media, publishing online content, or streaming videos.

Influencers get recognition from players that follow them. Consequently, game companies invest in influencers to endorse and perform marketing for their products (SJöBLOM; HAMARI, 2017). Hence, an influencer has direct relevance in such phenomena as viral marketing, innovation diffusion and behavior adoption (PEI; MORONE; MAKSE, 2018). To detect game influencers is therefore almost mandatory when looking for popular preferences, new trends and the like (SJöBLOM; HAMARI, 2017; HILVERT-BRUCE et al., 2018).

This chapter presents a novel framework to detect game influencers in SNGs; it is the second of the three main contributions of this MSc work. Given the actions of thousands of players in an online game, how to detect the game influencers? It is obviously necessary to model the players’ features. We express the problem as a classification task, and propose a player modeling heuristic based on temporal aspects of their games by analyzing the history of “likes” received over time. To validate our proposal, we studied the famous platform of video games Super Mario Maker (Nintendo, Kyoto, Japan), and report experimental results to show that our framework detects game influencers from different nations with high accuracy.

The next sections are organized as follows: we start with network formalization and prob-
4.1 Network and Problem

As described in Section 3.2 from Chapter 3, SMM allows the players to elaborate and share their own game levels. In this sense, this Social Network of Games includes a directed bipartite graph $G_{dev} = (V, E_{dev})$ with edges $e_{dev} = (p, c) \mid e_{dev} \in E_{dev}$ of players $p \in V$ sharing games $c \in V$. Additionally, graph $G_{dev}$ changes over time. In time $t_i$, graph $G_{dev,i}$ presents a determined topology, while in time $t_{i+1}$, where $t_i < t_{i+1}$, graph $G_{dev,i+1}$ may include new nodes, i.e., new players or games, and new edges. Therefore, since there exists a stream of graphs $[G_{dev,1}, G_{dev,2}, \ldots]$ as a function of time, it is a dynamic network (WESTABY, 2012).

Figure 20a illustrates an example network $G_{dev}$. The initial graph $G_{dev,i}$ has nodes $V = \{p_1, p_2, c_1, c_2, c_3\}$, i.e., two players and three games. The respective edges are $E_{dev} = \{(p_1, c_3), (p_2, c_1), (p_2, c_2)\}$; they define the games shared by each player. In $G_{dev,i+1}$, player $p_1$ shares a new game $c_4$. In $G_{dev,i+2}$, there appears a new player $p_3$ without sharing any game. Finally, in $G_{dev,i+3}$, nothing changes. Therefore, this network can only grow over time.

Besides creating new games, a player can also like games of other players in SMM platform. In this sense, SMM has also another directed bipartite graph $G_{like} = (V, E_{like})$ with edges $e_{like} = (p, c) \mid e_{like} \in E_{like}$ of players $p \in V$ that give “likes” to game levels $c \in V$. Note that the nodes in $V$ are the same for both graphs $G_{dev,i}$ and $G_{like,i}$, considering any time instant $t_i$. Similarly to $G_{dev}$, graph $G_{like}$ is also a dynamic network.
4.2 Related Work

Figure 20b illustrates an example network $G_{\text{like}}$. In time $t_i$, graph $G_{\text{like},i}$ has nodes $V = \{p_1, p_2, c_1, c_2, c_3\}$ and edges $E_{\text{like}} = \{(p_1, c_1), (p_2, c_3)\}$ indicating the players that liked game levels. In $G_{\text{like},i+1}$, player $p_1$ likes level $c_2$ and a new game $c_4$ appears with no likes. In $G_{\text{like},i+2}$, there appears a new player $p_3$ that likes $c_3$; also, player $p_2$ likes $c_4$. Finally, in $G_{\text{like},i+3}$, player $p_3$ likes games $c_2$ and $c_4$.

According to this discussion, SMM can be represented by two dynamic networks, $G_{\text{dev}}$ and $G_{\text{like}}$, whose topologies are correlated and evolve as a function of time. To analyze such temporal evolution is essential when spotting game influencers, since the games that they develop become popular right after they advertise them in other media, like Twitch and YouTube, while the other players’ games gain popularity gradually (ABISHEVA et al., 2014). The analysis of the evolution may therefore be even more important than looking for static graph topologies in individual time stamps. With that in mind, the main problem we tackle in this chapter is: how to spot game influencers by evaluating the temporal evolution of networks $G_{\text{dev}}$ and $G_{\text{like}}$?

As described in the previous Chapter 3, there exist other types of player-game interactions that lead to additional dynamic networks describing SMM, such as plays and time records. However, what influencers play and their broken records don’t really matter in this issue. Nevertheless, without loss of generality, we only consider $G_{\text{dev}}$ and $G_{\text{like}}$ in this chapter, and show that they are enough to accurately identify game influencers.

4.2 Related Work

This section describes existing algorithms to identify influential users in social networks. To the best of our knowledge, there is no research on the detection of influencers from SNG; the existing approaches focused on other types of social networks are therefore the closest related work. So, we outline the state-of-the-art in the following.

The naive approaches to detect digital influencers are based on the selection of the top-ranked nodes according to some centrality indices; more advanced strategies are greedy and heuristic methods (WANG et al., 2017; AL-GARADI et al., 2018). The centrality-based methods find a set of influential nodes based on the network topology (BARABÁSI; PÓSFAI, 2016). In Morone et al. (2016), it is presented an algorithm that ranks nodes based on collective influence propagation, which quantifies the nodes’ “influences” in the network. Meanwhile, Wang et al. (2017) infer influential nodes based on spreading paths, i.e., links.

Unfortunately, these algorithms work only on homogeneous networks, i.e., networks with a single type of node, while our problem refers to bipartite networks with nodes of players and games. Also, we need to correlate two dynamic networks and to analyze their temporal evolution. In this sense, it is difficult to use centrality indices because they infer knowledge based only on the topology of a single, static network. To tackle our problem, one must observe the nodes’ features, so to model game influencers’ features instead of ranking them.
In Chino et al. (2017), a dynamic graph of comments and online reviews was analyzed to detect suspicious users, e.g., spammers. The authors model the temporal aspect of the volume of activity, like the number of characters in a review, so to detect anomalies in the posts or comments. The proposed temporal aspect modeling is interesting and inspiring, but it does not tackle our problem because this approach works with a single dynamic network and uses specific application-related features that only make sense for text data. Besides, it detects suspicious users; not, influential users.

Other approaches extract features to describe social network users; then, they use the features to train Machine Learning (ML) algorithms to identify the most influential ones. ML-based approaches have great potential to identify influencers in large and complex networks (Al-Garadi et al., 2018). In Liu et al. (2014), temporal features are extracted based on microblog posts; they are named as post-features. Such features are then used to train ML algorithms that identify influential users in a temporal microblog network. The authors infer features for a user according to his/her number of followers, number of microblog posts, number of responses received and number of comments over time. Latter, Qi et al. (2015) extended the aforementioned work to identify domain-topics of influential users aimed at improving microblog search engines. They use temporal post-features combined with microblog text to identify the domain-topics of each influential user. Unfortunately, both works (Liu et al., 2014) and (Qi et al., 2015) use a single dynamic network and heavily depend on the post-features. Thus, it is not clear how to use them to tackle our problem. However, the methodology of extracting features to be given to ML-algorithms in order to spot influencers is inspiring; this idea is used as a basis in our work.

Finally, let us summarize the state-of-the-art on influencer detection. Some techniques use centrality metrics to rank influential users based on graph topology (Morone et al., 2016; Wang et al., 2017). Other methods are designed to work with dynamic networks; they commonly perform feature extraction on the network and on the users’ external contents to train ML algorithms (Liu et al., 2014; Qi et al., 2015). Also, user modeling based on temporal aspects has been applied (Chino et al., 2017). However, to the best of our knowledge, these works cannot tackle our problem mainly because they do not correlate more than one dynamic network or they fail to quantify their temporal evolution. In the next section, we present a novel proposal to detect game influencers in a large Social Network of Games.

4.3 Proposal

This section presents a new framework to detect influencers in Social Networks of Games. The main question here is: how to model the temporal evolution of the players’ actions on the dynamic networks $G_{dev}$ of developers and $G_{like}$ of likes? To answer this question, we propose a new player modeling heuristic based on temporal aspects of each game by extracting features from its history of “likes” over time and then correlating it with the respective maker.
Individually, neither $G_{dev}$ nor $G_{like}$ has intrinsic characteristics that are truly useful to spot influencers. For example, by using an outdegree centrality ranking on $G_{dev}$ we simply rank the players by the number of levels they share; note that it does not help much, since a user that shares many levels is not necessarily an influencer, while the opposite is also true. Similarly, $G_{like}$ would lead to a ranking of players by the number of likes given, which does not help much either. Both rankings fail to present player-related features that are relevant to identify influencers. Unfortunately, the same problem happens when using more elaborated centrality techniques, simply because there is not enough information in any of the networks alone.

As a consequence, one needs to capture relevant information from the temporal evolution of the networks. To make it possible, we propose to model this evolution using data streams; then, we extract relevant features from the streams and spot influencers using a classification ML algorithm. The next sections detail our proposal.

### 4.3.1 Stream Modeling

At first, we obtain from $G_{like}$ the quantity of likes received by each game $c$ up to each time instant $t_i$. It is done by computing the indegree centrality $g_{in}(c) \mid c \in V_i$ for each graph snapshot $G_{like,i} = (V_i,E_{like,i})$. Then, a game $c$ is represented by a stream of like-counting events $\text{likes}_c = [\epsilon_{c,1}, \epsilon_{c,2}, \ldots]$, where each event $\epsilon_{c,i}$ refers to a pair $(\text{likes}_{c,i}, t_i)$ with a number of likes $\text{likes}_{c,i}$ and a time stamp $t_i$. Let us illustrate this idea using our running example from Figure 20b. As can be seen in Figure 21a, the number of likes is constant over time for game $c_1$. Game $c_2$ receives one like in time $t_{i+1}$ and another one in time $t_{i+3}$; see Figure 21b. Figure 21c shows that game $c_3$ receives one like in time $t_{i+2}$. Finally, game $c_4$ receives a new like in each time instant posterior to its creation in time $t_{i+1}$; see Figure 21d.

In this way, we extract a like-counting stream through time instants $T = [t_1, t_2, \ldots]$ for each game. Additionally, for each player $p$, we analyze network $G_{dev}$ and extract a set $C_p = \{c_{p,1}, c_{p,2}, \ldots\}$ of games developed by the player. Therefore, for each player $p$ there is a set of his/her elaborated game levels and the corresponding like-counting streams over time.

The next step is to extract features that model each player appropriately. Note that a player $p$ develops games $c \in C_p$, and each game $c$ has a like-counting stream $\text{likes}_c$. The main
question here is: how to model the players based on interactions performed with their games? In order to accomplish this goal, we propose to extract temporal features from individual games, and then ensemble the features of games developed by each player to model the player. This process is detailed in the next section.

### 4.3.2 Extraction of Temporal Features

This section describes our strategy to extract temporal features from like-counting streams. We propose to use three distinct feature extractors, each one to capture a specific characteristic of the evolution of like counts. The extractors are described in the following.

#### 4.3.2.1 Extractor Linear Regression

The first extractor takes advantage of linear regression to infer features that describe game levels’ ascensions of likes, i.e., how the like counts grow over time. The linear regression model is widely used in social sciences; it is commonly applied to model the statistical relationship between variables (STRANG et al., 2016, p.25). We use the well-known Least Squares method to identify the linear function that best fits a like-counting ascension. Since each event $\varepsilon_{c,i}$ in the like-counting stream of a game $c$ has only two attributes, i.e., likes$_{c,i}$ and $t_i$, the best linear function can be summarized by a constant term $\alpha_c$ and a coefficient $\beta_c$; see Equation 4.1.

$$\text{likes}_{c,i} \simeq f_c(t_i) = \alpha_c + \beta_c \cdot t_i \tag{4.1}$$

In this sense, function $f_c(t_i)$ estimates the count of likes for a game $c$ in timestamp $t_i$, where the values of $\alpha_c$ and $\beta_c$ are inferred from stream $\text{likes}_c = [\varepsilon_{c,1}, \varepsilon_{c,2}, \ldots]$. Considering that $\overline{\text{likes}}_c$ is the average number of likes in the stream, and $\overline{t_c}$ is the corresponding average timestamp, the coefficients are defined in the following.

$$\beta_c = \frac{\sum t_i \cdot \text{likes}_{c,i} - \frac{1}{|\text{likes}_c|} \cdot \sum t_i \cdot \sum \text{likes}_{c,i}}{\sum t_i^2 - \frac{1}{|\text{likes}_c|} \cdot (\sum t_i)^2}$$
$$\alpha_c = \overline{\text{likes}}_c - \beta_c \cdot \overline{t_c}$$

Figure 22 illustrates the linear regression for an example game $c$ with a like-counting stream $\text{likes}_c = [(0,t_1), (1,t_2), (4,t_3), (6,t_4), (6,t_5), (7,t_6), (7,t_7)]$ collected during consecutive time instants between $t_1$ and $t_7$. Extractor LR finds the linear function that best fits the data; then, it extracts the angle $\angle_c = \tan^{-1}(\beta_c)$ and the coefficient of determination $R^2_c$ for a game $c$. In this example, $\angle_c = 51.34^\circ$ and $R^2_c = 0.88$. Angle $\angle_c$ represents how fast the number of likes received by a game $c$ grows over time. Meanwhile, the coefficient of determination $R^2_c$ is the proportion of the variance from the linear regression prediction. It ranges from 0 to 1, where $R^2_c = 1$ means that the like count growth over time defines exactly a line segment. We use coefficient $R^2_c$ to estimate a “curvature” on the stream; the lower is $R^2_c$, the larger is the “curvature”.
4.3. Proposal

Algorithm 1 is the pseudo-code for our extractor LR. Given a player \( p \) of interest, it builds in Line 10 a set \( C_p \) with all game levels developed by \( p \). Then, Lines 11 to 16 use linear regression to infer the linear function that best fits the stream of like counts \( \text{likes}_c \) of each game \( c \in C_p \) together with the corresponding angle \( \angle_c \) and coefficient of determination \( R^2_c \). As shown in Lines 9, 13, 14 and 15, the aforementioned procedure stores in sets \( \beta, \angle \) and \( R^2 \) the values \( \beta_c, \angle_c \) and \( R^2_c \) computed for each game \( c \). The next step is to use these metadata extracted from the games to infer features that describe their maker, i.e., player \( p \).

How to properly describe a player from the linear-regression-based metadata of his/her games? We propose to do it by extracting the following features: \( \angle_p, \overline{\angle}, \sigma_\angle, S_\angle, R^2_{\text{rank}} \) and \( \angle_{\text{rank}} \). Intuitively, feature \( \angle_p \) quantifies how “ascending” a player \( p \) is; it is computed in Line 17 as the angle \( \tan^{-1}(\text{mean}(\beta)) \) that corresponds to the mean slope of the games’ linear functions. Features \( \overline{\angle}, \sigma_\angle \) and \( S_\angle \) are respectively the mean, the standard deviation and the entropy of the angle coefficients \( \angle_c \) in set \( \angle \). They are computed in Lines 18, 19 and 20. Here, it is worth noting that entropy is a measure of randomness in information; it ranges from 0 to \( \infty \) as the randomness increases (Kamber; Han; Pei, 2012, ch. 8, p. 337). Therefore, our entropy-based feature \( S_\angle \) quantifies how similar are the “ascension patterns”, i.e., angles, observed in the numbers of likes received over time by games that a player \( p \) developed.

Finally, features \( R^2_{\text{rank}} \) and \( \angle_{\text{rank}} \) are in fact arrays with \( L \) values each, where \( L \in \mathbb{N}_{>0} \) is a user-defined input parameter. Note that \( L \) is the only parameter used in our proposal. As can be seen in Line 21, feature \( R^2_{\text{rank}} \) contains the \( L \) lowest coefficients of determination \( R^2_c \) in set \( R^2 \), so \( R^2_{\text{rank}} = [\min_1(R^2), \min_2(R^2), \ldots \min_L(R^2)] \). Similarly, feature \( \angle_{\text{rank}} \) has the \( L \) highest angles \( \angle_c \) in set \( \angle \), that is, \( \angle_{\text{rank}} = [\max_1(\angle), \max_2(\angle), \ldots \max_L(\angle)] \); see Line 22. Intuitively, features \( R^2_{\text{rank}} \) and \( \angle_{\text{rank}} \) capture extreme cases of games developed by player \( p \) with non-linear growth, i.e., high “curvature”, or fast ascension in their streams of like counts.
Algorithm 1 – Extractor Linear Regression

1: function LR-GAME(c) \hfill \triangleright \text{Game Modeling}
2: \hspace{1em} Build stream \text{likes}_c \text{ for game } c;
3: \hspace{1em} \alpha_c, \beta_c \leftarrow \text{LinearRegression} (\text{likes}_c);
4: \hspace{1em} \angle_c \leftarrow \tan^{-1}(\beta_c);
5: \hspace{1em} R^2_c \leftarrow R^2(\alpha_c, \beta_c, \text{likes}_c);
6: \hspace{1em} \text{return } \beta_c, \angle_c, R^2_c;
7: end function

8: function LR(p, L) \hfill \triangleright \text{Player Modeling}
9: \hspace{1em} \beta, \angle, R^2 \leftarrow \emptyset;
10: \hspace{1em} Build set \text{C}_p \text{ for player } p;
11: \hspace{1em} \text{for } c \in \text{C}_p \text{ do}
12: \hspace{2em} \beta_c, \angle_c, R^2_c \leftarrow \text{LR-GAME} (c);
13: \hspace{2em} \beta \leftarrow \beta \cup \{ \beta_c \};
14: \hspace{2em} \angle \leftarrow \angle \cup \{ \angle_c \};
15: \hspace{2em} R^2 \leftarrow R^2 \cup \{ R^2_c \};
16: \hspace{1em} \text{end for}
17: \hspace{1em} \angle_p \leftarrow \tan^{-1}(\text{mean}(\beta));
18: \hspace{1em} \angle \leftarrow \text{mean}(\angle);
19: \hspace{1em} \sigma_\angle \leftarrow \text{std}(\angle);
20: \hspace{1em} S_\angle \leftarrow \text{entropy}(\angle);
21: \hspace{1em} R^2_{\text{rank}} \leftarrow [\min_1(R^2), \min_2(R^2), \ldots, \min_L(R^2)];
22: \hspace{1em} \angle_{\text{rank}} \leftarrow [\max_1(\angle), \max_2(\angle), \ldots, \max_L(\angle)];
23: \hspace{1em} \text{return } \angle_p, \angle, \sigma_\angle, S_\angle, R^2_{\text{rank}}, \angle_{\text{rank}};
24: end function

4.3.2.2 Extractor Delta Rank

Delta Rank (DR) is the second feature extractor that we propose. While the previous extractor reports one global view on how the like counts of the game levels developed by a player grow over time, considering the whole temporal interval between the first measurement and the last one in each like-counting stream, the new extractor presents a higher-resolution view by describing the levels’ ascensions of likes between consecutive measurements in time. Intuitively, extractor DR infers features that describe quasi-instantaneous ascensions in like counts.

\( \Delta_{c,i} \in \mathbb{N} \) is the most basic information investigated here; it is defined in Equation 4.2. As can be seen, \( \Delta_{c,i} \) quantifies the difference between two consecutive measurements \( \text{likes}_{c,i} \) and \( \text{likes}_{c,i+1} \) in the like-counting stream \( \text{likes}_c \) of a game \( c \). In other words, \( \Delta_{c,i} \) stores how many new likes a game \( c \) received in the shortest known temporal interval \([t_i, t_{i+1}]\) initiated in time \( t_i \).

\[
\Delta_{c,i} = \text{likes}_{c,i+1} - \text{likes}_{c,i} \tag{4.2}
\]

Figure 23 illustrates this idea. It shows that the like counts of a fictitious game \( c \) have 6 pairs of sequential events, so \( \Delta_{c,1} = 1, \Delta_{c,2} = 3, \Delta_{c,3} = 3, \Delta_{c,4} = 0, \Delta_{c,5} = 1 \) and \( \Delta_{c,6} = 0 \). Let us consider that \( \Delta_c \) is the set of all \( \Delta_{c,i} \) values of a game \( c \). By ranking the values in \( \Delta_c \), it is
possible to identify the game’s greatest quasi-instantaneous ascension, which is \( \Delta_{c,2} = 3 \) in our running example. Also, one can compute the entropy \( S_{\Delta_c} \) of \( \Delta_c \) to quantify how random are the quasi-instantaneous ascensions of likes for game \( c \); in the example, it is \( S_{\Delta_c} = 1.33 \). The proposed extractor DR uses these values as a basis to infer features for a player \( p \) of interest.

Figure 23 – Delta values \( \Delta_{c,i} \) for a game \( c \)

Algorithm 2 is the pseudo-code for our extractor DR. Given a player \( p \) of interest, it builds in Line 13 a set \( C_p \) with all game levels developed by \( p \). Then, Lines 14 to 18 compute \( \Delta_c \) and \( S_{\Delta_c} \) for each game \( c \in C_p \). As is shown in Lines 12, 16 and 17, this procedure stores in set \( \Delta \) all values \( \Delta_{c,i} \) computed; similarly, set \( S \) receives the \( S_{\Delta_c} \) values. The next step is to use these metadata extracted from the games to infer features that describe their maker, i.e., player \( p \).

How to describe a player from his/her games’ quasi-instantaneous ascensions of likes? We propose to do it by extracting the following features: \( \bar{\Delta} \), \( \sigma_{\Delta} \), \( S_{\Delta} \), \( \Delta_{\text{rank}} \), \( \bar{S} \), \( \sigma_{S} \) and \( S_{\text{rank}} \). Features \( \bar{\Delta} \), \( \sigma_{\Delta} \) and \( S_{\Delta} \) are respectively the mean, the standard deviation and the entropy of set \( \Delta \); see Lines 19, 20 and 21. These features summarize with statistical measurements the general pattern seen in quasi-instantaneous ascensions of likes \( \Delta_{c,i} \) of games developed by player \( p \). Similarly, features \( \bar{S} \) and \( \sigma_{S} \) refer to the mean and the standard deviation of the entropy values \( S_{\Delta_c} \) in set \( S \); see Lines 23 and 24. These features quantify how random are the quasi-instantaneous ascensions of likes expected for a game of player \( p \). Finally, features \( \Delta_{\text{rank}} \) and \( S_{\text{rank}} \) are in fact arrays with \( L \) values each, where \( L \in \mathbb{N}_{>0} \) is a user-defined input parameter. Note that it is the same parameter used by the previous extractor LR. As can be seen in Line 22, feature \( \Delta_{\text{rank}} \) contains the \( L \) highest quasi-instantaneous ascensions of likes \( \Delta_{c,i} \) in set \( \Delta \), so \( \Delta_{\text{rank}} = \{\max_1(\Delta),\ \max_2(\Delta),\ldots\max_L(\Delta)\} \). Similarly, feature \( S_{\text{rank}} \) has the \( L \) highest entropy values \( S_{\Delta_c} \) in set \( S \), that is, \( S_{\text{rank}} = \{\max_1(S),\ \max_2(S),\ldots\max_L(S)\} \); see Line 25. Intuitively, features \( \Delta_{\text{rank}} \) and \( S_{\text{rank}} \) capture extreme cases of games developed by player \( p \) with very fast or quasi-random growth in their streams of like counts.

4.3.2.3 Extractor Coefficient of Angle

Coefficient of Angle (CA) is the third and last extractor that we propose. It also focuses on quasi-instantaneous ascensions of like counts; however, this time we consider angles of
Algorithm 2 – Extractor Delta Rank

1: function DR-GAME(c) ▷ Game Modeling
2: Build stream $\text{likes}_c$ for game $c$;
3: $\Delta_c \leftarrow \emptyset$;
4: for $i \leftarrow 1$ to $|\text{likes}_c| - 1$ do
5: $\Delta_{c,i} \leftarrow \text{likes}_{c,i+1} - \text{likes}_{c,i}$;
6: $\Delta_c \leftarrow \Delta_c \cup \{\Delta_{c,i}\}$;
7: end for
8: $S_\Delta \leftarrow \text{entropy}(\Delta_c)$;
9: return $\Delta_c, S_\Delta$;
10: end function

11: function DR(p, L) ▷ Player Modeling
12: $\Delta, S \leftarrow \emptyset$;
13: Build set $C_p$ for player $p$;
14: for $c \in C_p$ do
15: $\Delta_c, S_\Delta \leftarrow \text{DR-GAME}(c)$;
16: $\Delta \leftarrow \Delta \cup \Delta_c$;
17: $S \leftarrow S \cup \{S_\Delta\}$;
18: end for
19: $\bar{\Delta} \leftarrow \text{mean}(\Delta)$;
20: $\sigma_\Delta \leftarrow \text{std}(\Delta)$;
21: $S_\Delta \leftarrow \text{entropy}(\Delta)$;
22: $\Delta\text{rank} \leftarrow [\max_1(\Delta), \max_2(\Delta), \ldots \max_L(\Delta)]$;
23: $\bar{S} \leftarrow \text{mean}(S)$;
24: $\sigma_S \leftarrow \text{std}(S)$;
25: $S\text{rank} \leftarrow [\max_1(S), \max_2(S), \ldots \max_L(S)]$;
26: return $\bar{\Delta}, \sigma_\Delta, S_\Delta, \Delta\text{rank}, \bar{S}, \sigma_S, S\text{rank}$;
27: end function

ascension instead of plain like-counting differences to provide a distinct view on this aspect. $\theta_{c,i} \in \mathbb{R}$ is the most basic information investigated here; it is defined in Equation 4.3. As can be seen, $\theta_{c,i}$ quantifies the angle of ascension between two consecutive measurements $\text{likes}_{c,i}$ and $\text{likes}_{c,i+1}$ in the like-counting stream $\text{likes}_c$ of a game $c$. In this way, $\theta_{c,i}$ represents the angle of ascension in the shortest known temporal interval $[t_i, t_{i+1})$ initiated in time $t_i$.

$$\theta_{c,i} = \tan^{-1} \frac{\Delta_{c,i}}{t_{i+1} - t_i}$$

(4.3)

Figure 24 illustrates this idea for the same like-counting stream of a fictitious game $c$ used as an example in the previous section. Once again, we consider consecutive time instants between $t_1$ and $t_7$, thus $\theta_{c,1} = 45.0^\circ$, $\theta_{c,2} = 71.6^\circ$, $\theta_{c,3} = 63.4^\circ$, $\theta_{c,4} = 0.0^\circ$, $\theta_{c,5} = 45.0^\circ$ and $\theta_{c,6} = 0.0^\circ$. The angle measurements $\theta_{c,i}$ vary within interval $[0^\circ, 90^\circ)$; large values indicate high like-counting growth between two sequential events. Compared with the plain count of new likes $\Delta_{c,i}$ used in the previous extractor – see Figure 23, the angle values emphasize differences between intervals with few new likes, while they also attenuate differences when the counts
of new likes are elevated. For example, note that the absolute difference in plain count of new likes between intervals \([t_4,t_5]\) and \([t_5,t_6]\) is equal to the corresponding difference for \([t_2,t_3]\) and \([t_3,t_4]\), thus \(|\Delta_{c,4} - \Delta_{c,5}| = |\Delta_{c,2} - \Delta_{c,3}| = 1\). In spite of this fact, the corresponding angle-based differences are not the same, \(i.e., |\theta_{c,4} - \theta_{c,5}| > |\theta_{c,2} - \theta_{c,3}|\), since the counts of new likes are smaller in intervals \([t_4,t_5]\) and \([t_5,t_6]\) than their peers in intervals \([t_2,t_3]\) and \([t_3,t_4]\). Additionally, the angle-based measurement \(\theta_{c,i}\) takes into account the size of the interval analyzed, that is \(t_{i+1} - t_i\), as opposed to the corresponding count of new likes \(\Delta_{c,i}\). Due to the aforementioned distinctions between angles and plain counts of new likes, we propose to take advantage of one novel view on quasi-instantaneous game ascensions with extractor CA; it uses angle-based measurements \(\theta_{c,i}\) extracted from games to infer features that describe their maker.

Figure 24 – Coefficients of angle \(\theta_{c,i}\) for a game \(c\)

![Figure 24 – Coefficients of angle \(\theta_{c,i}\) for a game \(c\)](image)

Source: Elaborated by the author.

Algorithm 3 is the pseudo-code for our extractor CA. As shown in Lines 12, 15 and 16, the algorithm represents a player \(p\) of interest as the union of \(\theta_c = \{\theta_{c,1}, \theta_{c,2}, \ldots \theta_{c,|\text{likes}_c|-1}\}\) for each of his/her games \(c\), so that \(\theta = \theta_1 \cup \theta_2 \cup \ldots \theta_{|c_p|}\). Then, it analyzes set \(\theta\) to extract the following features: \(\bar{\theta}\), \(\sigma_\theta\) and \(\theta_{\text{rank}}\). The first two features are respectively the mean and the standard deviation of \(\theta_{c,i}\) values in set \(\theta\); see Lines 18 and 19. These features summarize with statistical measurements the general pattern seen in angle-based representations \(\theta_{c,i}\) of quasi-instantaneous like-counting ascensions occurring for games developed by player \(p\). Finally, feature \(\theta_{\text{rank}}\) is in fact an array with \(L\) values, where \(L \in \mathbb{N}_{\geq 0}\) is the same parameter used by the previous extractors. As can be seen in Line 20, \(\theta_{\text{rank}}\) contains the \(L\) highest angle values \(\theta_{c,i}\) in set \(\theta\); that is, \(\theta_{\text{rank}} = [\max_1(\theta), \max_2(\theta), \ldots \max_L(\theta)]\). Intuitively, \(\theta_{\text{rank}}\) captures extreme cases of games developed by player \(p\) with very fast growth in their streams of like counts, especially focused on like-counting ascensions that come right after temporal intervals with few new likes.

### 4.3.2.4 Ensemble

Each of the three feature extractors described so far infers a series of features for the players. They can be used individually, of course, but they may also be combined into a single feature extractor, as shown in Equation 4.4. Let us name the combined extractor as \(F_{\text{ALL}}\).

\[
F_{\text{ALL}}(p, L) = [LR(p, L), DR(p, L), CA(p, L)]
\]  \hspace{1cm} (4.4)
Chapter 4. Detecting Game Influencers

Algorithm 3 – Extractor Coefficient of Angle

1: function CA-GAME(c) ▷ Game Modeling
2: Build stream \( \text{likes}_c \) for game \( c \);
3: \( \theta_c \leftarrow \emptyset \);
4: for \( i \leftarrow 1 \) to \( |\text{likes}_c| - 1 \) do
5: \( \Delta_{c,i} \leftarrow \text{likes}_{c,i+1} - \text{likes}_{c,i} \);
6: \( \theta_{c,i} \leftarrow \tan^{-1}\frac{\Delta_{c,i}}{i_{i+1} - i_i} \);
7: \( \theta_c \leftarrow \theta_c \cup \{\theta_{c,i}\} \);
8: end for
9: return \( \theta_c \);
10: end function

11: function CA(p, \( L \)) ▷ Player Modeling
12: \( \theta \leftarrow \emptyset \);
13: Build set \( C_p \) for player \( p \);
14: for \( c \in C_p \) do
15: \( \theta_c \leftarrow \text{CA-GAME}(c) \);
16: \( \theta \leftarrow \theta \cup \theta_c \);
17: end for
18: \( \overline{\theta} \leftarrow \text{mean}(\theta) \);
19: \( \sigma_\theta \leftarrow \text{std}(\theta) \);
20: \( \theta_{\text{rank}} \leftarrow [\max_1(\theta), \max_2(\theta), \ldots \max_L(\theta)] \);
21: return \( \overline{\theta}, \sigma_\theta, \theta_{\text{rank}} \);
22: end function

Note that extractors LR, DR and CA share an input parameter \( L \in \mathbb{N}_{>0} \) that determines the quantity of ranked values to be returned. Each extractor has its own ranking particularities, as discussed before, but, whenever \( L \) is larger than the number of values available to be ranked, all extractors complete the rankings with zeros. Also, note that the extractors build independently a set of game levels \( C_p \) for each player \( p \) of interest, and like-counting streams \( \text{likes}_c \) for the games developed by \( p \). Obviously, the combined extractor \( F_{\text{ALL}} \) builds each \( C_p \) and each \( \text{like}_c \) only once; it does not create these structures redundantly for individual extractors.

4.4 Experimental Evaluation

This section discusses the experimental evaluation performed to validate our proposals. It also includes a brief description on the datasets studied and on the experimental setup. We aimed at answering the questions as follows:

Q1 How accurately can one identify game influencers by giving to a classification algorithm features from each one of the extractors that we propose?

Q2 What is the impact in accuracy when one uses the ensemble of feature extractors \( F_{\text{ALL}} \)?
4.4. Experimental Evaluation

**Q3** Are our feature extractors general enough to support the use of data from distinct nationalities for training and testing?

In the experiments, we studied 28 distinct classification algorithms. They are: AdaBoostClassifier, BaggingClassifier, BernoulliNB, CalibratedClassifierCV, ComplementNB, DecisionTreeClassifier, ExtraTreeClassifier, ExtraTreesClassifier, GaussianNB, GaussianProcessClassifier, GradientBoostingClassifier, KNeighborsClassifier, LinearDiscriminantAnalysis, LinearSVC, LogisticRegression, LogisticRegressionCV, MLPClassifier, MultinomialNB, NearestCentroid, NuSVC, PassiveAggressiveClassifier, Perceptron, QuadraticDiscriminantAnalysis, RandomForestClassifier, RidgeClassifier, RidgeClassifierCV, SGDClassifier and SVC.

The algorithms are implemented within the `sklearn` package (PEDREGOSA et al., 2011) in Python 3. They were all initially tested with their default input parameter values; later, we fine tuned the most accurate algorithms. For reproducibility, all codes, detailed results and algorithms used are freely available for download at GitHub\(^1\).

### 4.4.1 Datasets

We studied Canadian and French game makers from dataset SMMnet. As described in the previous Chapter 3, there are 74,915 game levels developed by 32,055 makers from Canada or France. They received more than 380 thousand likes from players of any nationality.

The data was split into datasets CA\(\text{N}\) and FRA. Dataset CA\(\text{N}\) includes the dynamic graphs \(G_{\text{dev}}\) and \(G_{\text{like}}\), referring to 34,479 games developed by Canadian makers; FRA has the corresponding graphs for 40,436 games elaborated by French makers. The idea is to use dataset CA\(\text{N}\) in an initial training/testing step, and FRA in a further validation step. To make it possible, we created ground truth metadata for CA\(\text{N}\). Specifically, we first ranked the top-100 Canadian makers based on the sum of likes received by their games. Then, we manually classified the top-100 makers as influencers or non-influencers by checking their social activities on the web. Our consensus is that influencers advertise their games in popular web sites of SMM communities. The web sites in the following were then investigated to build the ground truth: Reddit\(^2\), GameFAQs\(^3\), Twitch\(^4\), YouTube\(^5\), Facebook\(^6\), NintendoLife\(^7\) and Makers of Mario\(^8\). In total, 41 influencers were identified with this procedure; the remaining 59 players were considered to be non-influencers.

---

1 GitHub. <https://github.com/leomaurodesenv/paper-2019-iceis>.
2 Reddit. <https://www.reddit.com/r/MarioMaker/>
3 GameFAQs. <https://gamefaqs.gamespot.com/wii-u/805618-super-mario-maker>
4 Twitch. <https://www.twitch.tv>
5 YouTube. <https://youtube.com>
6 Facebook. <https://www.facebook.com>
7 NintendoLife. <http://www.nintendolife.com/forums>
8 Makers of Mario. <https://makersofmario.com/>
4.4.2 Evaluation of Individual Feature Extractors

In the first experiment, we aimed at answering question Q1: “How accurately can one identify game influencers by giving features from each proposed extractor to a classification algorithm?” To make it possible, we evaluated the feature extractors individually. Here, dataset CAN was used with a 5-fold cross-validation strategy. Note that we used $k = 5$ in the $k$-fold cross-validation due to the small number of labeled makers, i.e., 100. Extractors LR, DR and CA were evaluated with parameter $L$ ranging from 1 to 10.

Figures 25, 26 and 27 summarize the results for extractors LR, DR and CA, respectively. Each figure reports accuracy for the top-6 best classifiers, among the 28 techniques studied, when using features from one of our extractors. For the reader’s convenience, the results are shown both in linear plots and in heat map plots. As can be seen, the best accuracy for each extractor ranges from 67.0% to 74.0%; it was obtained with values between 8 and 10 for parameter $L$. Specifically, the best configuration per extractor was: (i) LR – best accuracy of 67.0% obtained by the DecisionTreeClassifier with 20 features, that is $L = 8$; (ii) DR – best accuracy of 69.0% obtained by the DecisionTreeClassifier when analyzing 25 features, i.e., $L = 10$, and; (iii) DA – 74.0% accuracy obtained by the GradientBoostingClassifier with only 11 features, so $L = 9$. For extractor DR, it is worth noting that the average accuracy of classifiers increased together with $L$. However, when $L$ becomes higher than the number of values available to be ranked, some features must receive zero values. In this sense, $L$ cannot be much greater than 10.

Figure 25 – Accuracy results for the top-6 best classifiers when using features from extractor LR. The input parameter $L$ was tuned with values from 1 to 10.

Source: Research data.
Figure 26 – Accuracy results for the top-6 best classifiers when using features from extractor DR. The input parameter $L$ was tuned with values from 1 to 10.

![Accuracy in linear plot](chart1)

(a) Accuracy in linear plot

(b) Accuracy in heat map plot

Source: Research data.

Figure 27 – Accuracy results for the top-6 best classifiers when using features from extractor CA. The input parameter $L$ was tuned with values from 1 to 10.

![Accuracy in linear plot](chart2)

(a) Accuracy in linear plot

(b) Accuracy in heat map plot

Source: Research data.
Table 6 reports detailed results for the top-3 best classifiers; besides accuracy, it includes measurements for the additional metrics precision, recall and F1-score. As can be seen, the same patterns discussed in the previous paragraph about the values of accuracy are also apparent in the additional metrics of evaluation. In summary, our feature extractors LR, DR and CA presented average results for a classification task, with approximately 70% accuracy, 60% precision, 60% recall and 60% F1-score. So far, let us highlight extractor CA; compared with the other extractors, it created less attributes and still reached the highest performance.

Table 6 – Experiments: evaluation of the top-3 best classifiers in each experiment

| Algorithm               | Accuracy   | Precision | Recall     | F1-score   |
|-------------------------|------------|-----------|------------|------------|
| **LR**                  |            |           |            |            |
| DecisionTreeClassifier  | 0.670 ± 0.079 | 0.621 ± 0.109 | 0.645 ± 0.100 | 0.595 ± 0.100 |
| BernoulliNB             | 0.665 ± 0.106 | 0.432 ± 0.145 | 0.600 ± 0.100 | 0.489 ± 0.131 |
| QuadraticDiscriminantAnalysis | 0.665 ± 0.106 | 0.432 ± 0.145 | 0.600 ± 0.100 | 0.489 ± 0.131 |
| **DR**                  |            |           |            |            |
| DecisionTreeClassifier  | 0.690 ± 0.063 | 0.680 ± 0.089 | 0.658 ± 0.070 | 0.632 ± 0.073 |
| ExtraTreeClassifier     | 0.670 ± 0.067 | 0.616 ± 0.101 | 0.653 ± 0.075 | 0.607 ± 0.087 |
| GradientBoostingClassifier | 0.670 ± 0.055 | 0.593 ± 0.102 | 0.613 ± 0.058 | 0.579 ± 0.077 |
| **CA**                  |            |           |            |            |
| GradientBoostingClassifier | 0.740 ± 0.069 | 0.745 ± 0.068 | 0.766 ± 0.076 | 0.722 ± 0.072 |
| BaggingClassifier       | 0.696 ± 0.090 | 0.648 ± 0.104 | 0.675 ± 0.076 | 0.640 ± 0.096 |
| ExtraTreeClassifier     | 0.680 ± 0.071 | 0.650 ± 0.086 | 0.687 ± 0.091 | 0.644 ± 0.079 |
| **F_{ALL}**             |            |           |            |            |
| LogisticRegression      | 0.808 ± 0.108 | 0.808 ± 0.108 | 0.733 ± 0.157 | 0.745 ± 0.142 |
| RidgeClassifierCV       | 0.775 ± 0.105 | 0.733 ± 0.131 | 0.675 ± 0.155 | 0.685 ± 0.144 |
| LinearSVC               | 0.750 ± 0.113 | 0.750 ± 0.111 | 0.683 ± 0.154 | 0.688 ± 0.139 |
| **F_{ALL} Tuning**      |            |           |            |            |
| LogisticRegression      | 0.8709 ± 0.066 | 0.9029 ± 0.053 | 0.8588 ± 0.080 | 0.8573 ± 0.076 |
| RidgeClassifierCV       | 0.8064 ± 0.220 | 0.7903 ± 0.270 | 0.8225 ± 0.212 | 0.7849 ± 0.255 |
| LinearSVC               | 0.8387 ± 0.094 | 0.8596 ± 0.097 | 0.8306 ± 0.103 | 0.8266 ± 0.099 |

Source: Research data.

### 4.4.3 Ensemble Experiment

In the second experiment, we aimed at answering question Q2: “What is the impact in accuracy when one uses the ensemble of feature extractors $F_{ALL}$?” To make it possible, we analyzed once more the dataset CAN using a 5-fold cross-validation strategy. Figure 28 presents considerably better results compared with the individual extractors. With that in mind, we further investigated how to improve these results by performing an exhaustive search through the hyperparameter space of each classifier. This exhaustive search cost 2.666 experiments to improve tree classifiers. For LogisticRegression, the fine tuning was performed by testing several values for parameters $C, dual, fit\_intercept, max\_iter, penalty$ and $solver$. The same procedure was also followed for the other two classifiers: parameters $alphas, cv, fit\_intercept$.
and `store_cv_values` were fine tuned for `RidgeClassifierCV`, while `LinearSVC` was exhaustively tested with distinct values for parameters `C`, `dual`, `fit_intercept`, `loss`, `max_iter` and `penalty`. As can be seen in Table 6, the parameter tuning allowed us to obtain 87.1% accuracy, 90.3% precision, 85.9% recall and 85.7% F1-score with classifier `LogisticRegression`.

For the purpose of reproducibility, we report as follows the best configuration identified for each classifier: (i) `LogisticRegression`, `C = 0.9`, `dual = True`, `fit_intercept = True`, `max_iter = 100`, `penalty = l2` and `solver = warn`; (ii) `RidgeClassifierCV`, `alphas = [0.1, 1.0, 10.0]`, `cv = 7`, `fit_intercept = True` and `store_cv_values = False`, and; (iii) `LinearSVC`, `C = 0.4`, `dual = True`, `fit_intercept = False`, `loss = squared_hinge`, `max_iter = 100` and `penalty = l2`.

Figure 28 – Accuracy results for the top-6 best classifiers when using features from the ensemble of extractors `F_ALL`. The input parameter `L` was tuned with values from 1 to 10.

4.4.4 Validation

In the third and last experiment performed, we aimed at answering question Q3: “Are our feature extractors general enough to support the use of data from distinct nationalities for training and testing?” To make it possible, we trained a classifier with data from Canadian game makers and tested it with French makers. The best configuration identified so far was used in this experiment; that is, extractor `F_ALL` with `L = 9` and classifier `LogisticRegression` with the fine tuned parameter values reported in Section 4.4.3.

The validation experiment evaluated the generality of our framework. Specifically, we took the whole dataset `CAN` for training and dataset `FRA` for testing. At first, we ranked the top-100
game makers from dataset FRA based on their total number of likes received. Then, we used extractor $F_{ALL}$ to infer features for these players and the classifier to label the game influencers. Thanks to our proposed framework, the algorithm automatically labeled 27 players as game influencers, from which 21 were manually confirmed as true influencers by following the same ground-truth-creation procedure described in Section 4.4.1. Thus, our proposed model presented 77.8% precision to find influencers using a classifier trained and tested with data from distinct countries. These results indicate that our proposal is generic enough to accurately model the behaviour of game influencers, even when they have different nationalities.

4.5 Discussion

In this chapter, we presented our proposed strategy to detect game influencers in a Social Network of Games. It is the second main contribution of this MSc work. The central question investigated here was: given the actions of thousands of players in an online platform of video games, how to detect the most influential players? It was obviously necessary to model the general behavior of each player, but how could it be done? In order to tackle the problem, we proposed to capture relevant information from two correlated dynamic networks over time, which could not be performed with the existing related work. Then, we described how to model its evolution using data streams, from which we extracted relevant features that properly represent each player’s behavior, and mapped the game-influencer-detection problem into a classification task that uses our features as input. Finally, we validated our proposal by studying the famous game Super Mario Maker from Nintendo, Kyoto, Japan.

Our novel framework includes three feature extractors: LR, DR and CA. They are unsupervised and model the temporal aspects of the players’ actions on the social network. In the experimental evaluation, 28 classification algorithms were studied. Using our features as input, the LogisticRegression classifier obtained the best results with 87.1% accuracy, 90.3% precision, 85.9% recall and 85.7% F1-score. We also demonstrated that the new framework is generic enough to infer features that accurately highlight influencers in spite of their nationalities.

Finally, it is interesting to note that, in theory, our methodology can also be used for non-game-related applications, such as to spot influencers in other types of social networks with similar characteristics by analyzing the “likes” received by posts over time. Minor adaptations may be necessary though, due to domain specificities, so we consider it to be a promising topic for future work. Basically, two dynamic networks are the only requirement; one to store the creator of digital content and another to represent positive reactions of other users to the content, e.g., “likes”, as described in this chapter. Other possibility for future work is to apply regression modeling instead of classification so to study the players’ different degrees of influence over other players.
CHAPTER 5

CHARACTERIZING POPULAR GAMES

Advances in Data Mining (DM) and Machine Learning (ML) are increasing in the number of systems designed to help humans to express their creativity with art, music, education, games and the like. DM techniques support humans to develop new games; they are especially useful in game design (YANNAKAKIS; TOGELIUS, 2018); techniques and algorithms that assist in the selection of game objects for the elaboration of a level. Game design is a multidisciplinary activity that mixes technical and aesthetic aspects (SALEN; TEKINBAŞ; ZIMMERMAN, 2004). Due to design and technical challenges, game designers seek for systems and methods to make the process of creation easier (GUZDIAL; RIEDL, 2018).

This context has created a new area of research known as Game Artificial Intelligence, or Game AI for short. Game AI is a generic term that refers to the application of DM techniques in the game context; it goes from simple pattern discovery in games to the use of DM algorithms to automatically generate new game levels. Therefore, Game AI can be seen as a source of inspiration for game designers, in terms of encouraging them to rethink a design or to come up with new ideas (YANNAKAKIS; TOGELIUS, 2018; GUZDIAL et al., 2019).

The many existing games are classified by genre, such as games of platform (YANNAKAKIS; TOGELIUS, 2018). A game of platform is a digital game where the main character runs and jumps between platforms and obstacles, challenging enemies and collecting bonus items, e.g., in the games of the Super Mario Bros series, from Nintendo, Kyoto, Japan. Popular games of platform like the aforementioned one are among the most successful digital games ever developed (YANNAKAKIS; TOGELIUS, 2015).

When elaborating a new game of platform, the designers have to pick the items and characters that will appear in each game level. Usually, there exist thousands of possible combinations that can be chosen among a set of predefined items and characters. Nevertheless, it is commonly unclear what combinations are the best ones to attract the players, thus making the new game successful. Given the many possible items and characters to be used in a new game
level, how to select best option? To help answering the question, we present in this chapter a new framework designed to uncover patterns among the combinations of items and characters that were previously used in successful games of platform. Thanks to our framework, we were able to uncover and report combinations that are commonly present in popular levels developed by users of the game Super Mario Maker, from Nintendo, Kyoto, Japan. Both the framework and the novel patterns uncovered form the third and last central contribution of this MSc work.

This chapter is organized as follows. The main related work is described in Section 5.1. Then, our proposed methodology, the case study performed and the results obtained are discussed in Sections 5.2 and 5.3. Finally, Section 5.4 concludes the chapter.

5.1 Related Work

Automated tools have been used to support game design (YANNAKAKIS; TOGELIUS, 2018; GUZDIAL et al., 2019). For example, Guzdial and Riedl (2018) present a novel ML algorithm that extracts patterns from existing games to produce new ones, so it creates new games that contain characteristics of multiple other games. In this way, the algorithm produces novel output in comparison with its input. However, this algorithm requires a huge dataset of existing games and a description of the desired output in the training process. Thus, it heavily depends on the input received, which limits the creativity of the technique. Hence, this algorithm replies game design based on its experience, i.e., the input, instead of offering new options.

Guzdial et al. (2019) developed the Morai Maker AI system to help game designers in building new levels for a game of platform that is very similar to SMM. The development is performed in collaboration with the system itself, in such a way that the designer starts the creative process and alternates with the AI to make changes within the same environment of edition. Experiments indicate that the AI leads the game designers to modify their practices; and game designers perceived that the AI has a great potential value to support their work. Nevertheless, the system does not consider any characteristic of popular games. It only takes the user design as input for training and replies his/her style, which makes Morai Maker unable to create new patterns. Thereby, it is only appropriate for expert designers.

Beaupre et al. (2018) present a novel pattern-based algorithm able to automatically generate game levels by reproducing patterns that are common to other levels. It extracts features from known levels to generalize and replicate their most common items. This technique uses an optimization strategy with a fitness function to monitor the quality of the new levels produced. Unfortunately, the literature indicates that it is usually impractical to evaluate the quality of new content without a specialist (BROWNE; LIAPIS; WINANDS, 2019; YANNAKAKIS; TOGELIUS, 2018). Also, this work was designed for action and labyrinth games, not for games of platform. Nevertheless, their design patterns are interesting and very inspiring.

Other researchers focus on automated level design for shooting games. For example,
Karavolos, Liapis and Yannakakis (2018) present a system that uses simulated, two-player gameplays for training. The authors were interested in discovering what items of a level would result in a balanced match given a set of avatar classes, so they considered diverse game levels and different avatar classes. Unfortunately, this work was developed for shooting games, not for games of platform. The game items studied are very distinct from those we are interested in.

Games also have a huge potential to support education by including entertainment into the learning process (Bennis; Benhlima, 2015; Classe et al., 2019). Such kind of games are commonly known as serious games. In this context, Classe et al. (2019) developed a new strategy to help humans when designing serious games. Their contribution is a methodology based on business processes that guides the game designer by offering support in tasks where his/her creativity must take place. Unfortunately, the game designer does not know beforehand what game items may be used, which would be helpful during the creative process.

In summary, many related work focus on automatic game design (Guzdial; Riedl, 2018; Beaupre et al., 2018; Karavolos; Liapis; Yannakakis, 2018). However, it is challenging to evaluate the quality of the new content generated (Browne; Liapis; Winands, 2019; Yannakakis; Togelius, 2018). In this sense, others focus on semiautomatic design by developing systems to help humans during the creative process; not, to substitute the game designer (Guzdial et al., 2019). Nevertheless, some researchers highlight the need for game design that is based on player engagement as well as on exploratory case studies (Browne; Liapis; Winands, 2019). Inspired by Beaupre et al. (2018), we present in the following a novel framework to discover general design patterns that are common in popular games of platform. Specifically, we focus on identifying items and characters that frequently appear in game levels that attracted high engagement from players, i.e., game levels with many “likes”.

5.2 Extracting Features from Popular Games

This section presents our proposed methodology to describe successful games of platform. We are particularly interested in extracting numerical features to represent game levels by the set of items and characters that they contain. Figure 29 summarizes our approach. At first, we propose to filter out levels that were created by game influencers so to avoid bias in the analysis. Then, we collect data from the remaining levels and extract features that describe their items and characters. At last, we propose to aggregate similar features into general, high-level features so to both suit our framework to any game of platform and reduce the data sparsity and dimensionality. In a case study, we propose to investigate the famous game of platform Super Mario Maker that was developed by Nintendo, Kyoto, Japan.
Chapter 5. Characterizing Popular Games

5.2.1 Player Filtering

As discussed in Chapter 4, game influencers are players with elevated potential to influence others. They commonly produce digital content, e.g., online videos and posts in social media, that attract the attention of other players when looking for entertainment. In this sense, game influencers may receive recognition in their levels due to the popularity of other types of digital content that they produce, and not necessarily due to positive aspects of the levels themselves. Therefore, when characterizing successful game levels, we propose to ignore the levels developed by game influencers so to avoid bias in the process of analysis. We only consider popular levels elaborated by non-influencer makers.

To make it possible, we took advantage of previous contributions of this MSc work; that is, dataset SMMnet from Chapter 3 and the novel framework to detect game influencers described in Chapter 4. At first, we ranked the top-100 Canadian makers and the top-100 makers from France based on the sum of likes received by all of the game levels that they developed. Then, we used our framework to identify game influencers. There were 41 influencers and 59 non-influencers among the top-100 most successful Canadian makers, while 27 influencers and 73 non-influencers were identified among the French makers. Note that one non-influencer from France deleted his/her account, so only 72 non-influencers were considered in our case study.

Even the most successful makers may have created game levels that did not attract much attention. With that in mind, we decided to select for our analysis only the most successful levels developed by the Canadian and French makers identified in the previous step. The 59 non-influencers selected from Canada have developed a total of 760 game levels. Unfortunately, some of these levels had been deleted from the system, so we had to ignore them in the case study. Note that a game level can be removed by the user to make room for new levels, or by Nintendo, due to irregularities or non-popularity. The remaining levels were then sorted based on the sum of likes that they received, and the top-360 levels with the highest numbers of likes were selected for the analysis. Meanwhile, the 72 French makers elaborated a total of 1,984 levels, from which we selected for the case study the top-360 most successful ones in a similar manner.

In this way, there are selected 720 popular levels that were produced by successful makers from Canada or France, also excluding the bias that game influencers may generate. In
5.2. Extracting Features from Popular Games

the next section, we describe the data collection phase.

5.2.2 Data Collection

As described before, our main goal in this chapter is to uncover common design patterns from popular games of platform. So far, there are identified 720 popular game levels from one game of platform that is well-known worldwide, and we want to characterize them in a case study. To make it possible, we must have access to detailed descriptions about the design of these levels, including information on the items and characters that they contain. However, such descriptions are not present in our dataset SMMnet. It is important to remember that SMMnet stores only data about game usage, which was crawled from a public website; due to obvious copyright-related reasons, it does not include the games themselves. As described in Chapter 3, SMMnet stores basic information about the player that created a game level and metadata on how the level was used, including the number of times that it was played, the corresponding world time record, the number of likes received, and the like, but not descriptions about the levels themselves. To the best of our knowledge, such descriptions can only be obtained by downloading the levels in a video game console, not in a computer, for posterior analysis.

Due to the aforementioned reason, we manually downloaded the 720 game levels selected from SMM using one Nintendo Wii U console. Note that these levels have a considerable diversity; they were elaborated by players of two distinct nationalities, and represent four different game styles as well as four degrees of difficulty. Specifically, when grouping the game levels by their style, there are: 244 (33.89%) levels that follow style Super Mario Bros; 225 (31.25%) levels that follow Super Mario Bros U; 162 (22.50%) levels that follow Super Mario World, and; 89 (12.36%) levels that follow Super Mario Bros 3. In a similar manner, when grouping the levels by their difficult, there are: 147 (20.42%) levels that are considered to be easy; 286 (39.72%) are normal levels; 146 (20.28%) levels are in the expert degree of difficulty, and; 141 (19.58%) levels are in the super expert degree. Finally, there are exactly 360 game levels developed by players of each one of the two nationalities considered. Thereby, the game levels are perfectly balanced according to the nationality of their makers. Also, their game styles are nearly balanced, except for style Super Mario Bros 3 that represents only 12.36% of the data collection. The game difficulties are quasi-balanced too; the main exception is the set of normal-difficulty levels that represents nearly 40% of the data. In addition to the considerable data diversity, we argue that a collection of 720 game levels constitutes one substantial amount of data that is enough support our proposed case study. The next section describes the phase of feature extraction.

5.2.3 Feature Extraction

Once the 720 game levels of interest had been available, we needed to extract relevant features that represent their main characteristics. To make it possible, we propose to extract a histogram of items and characters. That is, we propose to represent a game level of interest
with features that store counts of occurrences for each possible type of item or character available in SMM. For example, a feature $F_{\text{NoteBlock}}$ counts how many items of type “Note Block” are present at each game level of interest. In a similar manner, a feature $F_{\text{PiranhaPlant}}$ stores occurrence counts for characters of type “PiranhaPlant”; that is, opponents of this particular type.

In order to extract the histogram from the binary file of each game level, we developed a novel tool by means of reverse-engineering and careful examination of the binaries aimed at identifying the items and characters that are available for use at SMM. This process led us to discover that there are 173 unique items or characters in the game; note that we ignored non-interactable items, such grounding items and ornaments. Appendix A reports the list of all 173 features that we propose to extract. Note that our feature extraction tool is available for download at the npm repository\(^1\). In this sense, we extracted histograms of items and characters from all of the 720 levels downloaded in the previous phase. In total, these levels include more than 310 thousand items or characters. The next section describes the phase of feature aggregation.

### 5.2.4 Feature Aggregation

This section presents our proposal for feature aggregation. We intend to aggregate similar features from the previous phase into general, high-level features so to both suit our framework to any game of platform and reduce the data sparsity and dimensionality. Each aggregated feature represents one generic game object; that is, it corresponds to a set of similar items or characters available for the designer when elaborating/drawing a level. In the literature, other researchers have previously performed feature aggregation. Let us highlight the work of Beaupre et al. (2018) that propose five aggregated features: (1) Collectable, for collectible items; (2) Harmful, for opponents that may cause damage; (3) Solid, for walls, etc.; (4) Avatar, for the players’ characters

---

\(^1\) Feature extraction. <https://www.npmjs.com/package/smm-course-viewer> (accessed August 1, 2019).
5.2. Extracting Features from Popular Games

in the game, and; (5) Other, for anything else. In this MSc work, we are particularly interested in describing interactable items and characters of general games of platform, so we must focus on features Collectable and Harmful. However, the use of only two aggregated features would not help game designers much; it would be excessively abstract, since games of platform commonly contain several subtypes of items and characters. Therefore, we propose to expand this process.

Figure 30 illustrates our proposal for feature aggregation. In our modeling, every aggregated feature refers to the sum of original features that count occurrences of similar items or characters. For example, feature $F_{\langle I, \{\text{vehicle}\} \rangle}$ aggregates the occurrence counts of all vehicle items into a single, generic count. That is: $F_{\langle I, \{\text{vehicle}\} \rangle} = F_{\text{LakituCloud}} + F_{\text{KoopaClown}} + F_{\text{FireKoopaClown}} + F_{\text{YoshiEgg}}$, where $F_{\text{LakituCloud}}$, $F_{\text{KoopaClown}}$, $F_{\text{FireKoopaClown}}$ and $F_{\text{YoshiEgg}}$ are features extracted in the previous phase. Appendix A details every one of the aggregations that we propose. In a nutshell, our modeling has two main categories: Items and Characters. They are both divided into subcategories, as we describe in the following.

Items can be either collectibles, such as power-ups and bonus items, or auxiliary items, like puzzle and teleportation items. We defined 8 aggregated features for item subcategories:

1. $F_{\langle I, \{\text{powerUp}\} \rangle}$: count of items that enhance some avatar’s characteristics;
2. $F_{\langle I, \{\text{vehicle}\} \rangle}$: count of items for transportation;
3. $F_{\langle I, \{\text{support}\} \rangle}$: count of items that support the player, such as blocks, or check point items that allow saving the game state;
4. $F_{\langle I, \{\text{jump}\} \rangle}$: count of items that help the player to jump;
5. $F_{\langle I, \{\text{move}\} \rangle}$: count of items that move the player, e.g., a horizontal lift;
6. $F_{\langle I, \{\text{teleport}\} \rangle}$: count of items for teleportation;
7. $F_{\langle I, \{\text{puzzle}\} \rangle}$: count of items that activate other items, like a key for a door;
8. $F_{\langle I, \{\text{bonus}\} \rangle}$: count of scoring items, such as a coin.

Category Characters aggregates features that count the occurrence of harmful opponents, like monsters and spikes. We identified four aspects to generally represent an opponent: (i) Vitality – opponents that can be defeated without using any special item are considered to be fragile, while robust opponents are immortal or can only be defeated by using a power-up item; (ii) Move – opponents can either walk, fly or be stable; (iii) Ability: opponents can be shooters or non-shooters, and; (iv) Size: opponents can have a normal or a big size. We propose to use this categorization in order to aggregate features that describe similar opponents. It leads to 24 possible subcategories of characters, so 24 features. For example, feature $F_{\langle C, \{\text{fragile, walk, non-shooter, normal}\} \rangle}$ stores occurrence counts for normal-sized, non-shooter
opponents that are fragile and walk. Here, we assume that the default ability and size are respectively non-shooter and normal. This allows to summarize our notation; for example, it allows to specify the aforementioned feature in a shorter version, that is $F_{<C,\{\text{frail}, \text{walk}\}}$.

In summary, we aggregate the 173 features received from the previous phase into 32 high-level features so to both suit our framework to any game of platform and reduce the data sparsity and dimensionality. In our case study, these features describe 720 popular levels of SMM that were produced by successful makers from Canada or France, also excluding the bias that game influencers may generate. Our new feature-aggregation strategy is illustrated in Figure 30, and the details are given in Appendix A. As opposed to the related work, it considers subcategories of interactable items and characters to present a more realistic scenario for game designers. There are features for 8 subcategories of items and 24 subcategories of characters, which leads to billions of object combinations that can appear in a game level. The next section presents insights on the combinations of items and characters used to draw the 720 popular SMM levels investigated in our case study. We are especially interested in discovering the most frequent and useful combinations, since they may guide designers when developing new games of platform.

5.3 Characterizing Popular Games

At this point, there are found 720 popular levels from a game of platform that is well-known worldwide, and extracted 32 features from them by counting occurrences of similar items and characters. Thus, each level is represented by the histogram of categorized game objects that it contains. Now, we aim to spot combinations of objects that commonly occur on these levels. To make it possible, we propose a subspace-clustering-based analysis followed by the discovery of association rules on the clusters. The next sections detail our proposal and the results obtained.

5.3.1 Analyzing Popular Games with Subspace Clustering

As discussed in Chapter 2, subspace clustering techniques identify clusters of elements that have similar values on subsets of features. In our context, these techniques can spot clusters of game levels that share similar occurrence counts with regard to subsets of categorized game objects. As a consequence, they allow to identify the combinations of items and characters that most frequently occur in the 720 popular levels selected for our case study.

Figures 31 and 32 illustrate the main steps of our proposal. At first, we use a subspace clustering algorithm to analyze the 32 features that were previously extracted; it allows to detect clusters that are only dense when projected into subspaces of the original feature space. For example, let us consider the toy clusters illustrated in Figure 31. Cluster $G_1$ includes six game levels that are close to each other only when they are projected into feature $F_{<I,\{\text{vehicle}\}}$, so $G_1$ exists in the 1-dimensional subspace formed by this feature. Similarly, cluster $G_2$ includes seven game levels and it is only dense in the subspace of feature $F_{<C,\{\text{robust, stable}\}}$. Finally,
5.3. Characterizing Popular Games

Cluster $G_3$ includes four game levels that are very similar to each other in the two-dimensional subspace formed by features $F_{<I,\{\text{vehicle}\}>}$ and $F_{<C,\{\text{robust,stable}\}>}$. Once the clusters are found, we propose to filter out irrelevant clusters like the ones created due to default configurations that the SMM system makes to every new game level. In a final phase, we analyze the remaining clusters so to understand their semantics. The next sections describe our proposal in detail.

Figure 31 – Spotting subspace clusters

Figure 32 – Pruning and analyzing

(1) Subspace clustering

- $G_1 = \{ <F_{<I,\{\text{vehicle}\}>},[4,6]> \}$
- $G_2 = \{ <F_{<C,\{\text{robust,stable}\}>},[10,15]> \}$
- $G_3 = \{ <F_{<I,\{\text{vehicle}\}>},[4,6]>, <F_{<C,\{\text{robust,stable}\}>},[10,15]> \}$

(2) Pruning

- $G_1 = \{ <F_{<I,\{\text{vehicle}\}>},[0,1]> \}$
- $G_2 = \{ <F_{<C,\{\text{robust,stable}\}>},[0,2]> \}$
- $G_3 = \{ <F_{<I,\{\text{vehicle}\}>},[4,6]>, <F_{<C,\{\text{robust,stable}\}>},[10,15]> \}$
- $G_4 = \{ <F_{<I,\{\text{vehicle}\}>},[0,0]>, <F_{<C,\{\text{fragile,walk}\}>},[0,8]> \}$
- $G_5 = \{ <F_{<C,\{\text{robust,stable}\}>},[10,15]>, <F_{<C,\{\text{fragile,walk}\}>},[0,8]> \}$

(3) Analysis

- $G_3 = \{ <F_{<I,\{\text{vehicle}\}>},[4,6]>, <F_{<C,\{\text{robust,stable}\}>},[10,15]> \}$
- $G_5 = \{ <F_{<C,\{\text{robust,stable}\}>},[10,15]>, <F_{<C,\{\text{fragile,walk}\}>},[0,8]> \}$

Source: Elaborated by the author.

5.3.1.1 Spotting Clusters in Subspaces

For this phase, we selected three well-known subspace clustering techniques: CLIQUE, P3C and STATPC. The default parameter configuration was chosen for each technique. CLIQUE was tested with values 10 and 0.5 for its parameters $\xi$ and $\tau$, respectively. P3C was tuned with value $1.0E - 4$ for its Poisson threshold. STATPC used $\alpha_0 = 1.0E - 10$ and $\alpha_K = \alpha_H = 0.001$. Unfortunately, the last two algorithms failed to obtain appropriate results from our dataset. P3C did not return any cluster. STATPC identified 10 overlapping clusters that refer to the full, 32-dimensional feature space, not to subspaces. Thus, they both failed to identify relationships that refer to subsets of features or poorly partitioned the game levels. Following a suggestion from its original paper, we also tested STATPC with values $\alpha_0 \in \{1.0E - 03, 1.0E - 05, 1.0E - 10, 1.0E - 15, 1.0E - 20\}$, $\alpha_K \in \{1.0E - 03, 1.0E - 05, 1.0E - 10, 1.0E - 15, 1.0E - 20\}$ and $\alpha_H \in \{1.0E - 03, 1.0E - 05, 1.0E - 10, 1.0E - 15, 1.0E - 20\}$, but the results did not change. We believe that it happened because both P3C and STATPC depend on probability-based tests that work best...
with thousands of instances or more, while there are 720 game levels to analyze. Additionally, there is a considerable sparsity in our dataset: nearly 57% of the occurrence counts are zeros. It might have impaired these algorithms, since many game levels look similar due to the elevated presence of zeros.

On the other hand, CLIQUE inferred a very large number of clusters; that is, 3,573 clusters. They exist in subspaces whose dimensionalities go from 1 up to 9. Table 7 reports the number of clusters identified versus the dimensionality of their corresponding subspaces. Note that it would be a very exhaustive task to manually analyze all clusters. It would also be probably meaningless, especially because many clusters may have been created due to default configurations that the SMM system makes to every new game level. Thus, we perform one pruning procedure to filter out irrelevant clusters automatically. It is described in the next section.

| Dimensionality of the subspace | Number of clusters |
|-------------------------------|--------------------|
| 1                             | 16                 |
| 2                             | 180                |
| 3                             | 554                |
| 4                             | 945                |
| 5                             | 955                |
| 6                             | 620                |
| 7                             | 168                |
| 8                             | 103                |
| 9                             | 32                 |
|                               | Total: 3,573       |

Source: Research data.

**5.3.1.2 Pruning Irrelevant Clusters**

In this section, we present a heuristic to filter irrelevant clusters out. It its illustrated in Figure 32. This phase is mainly motivated by the fact that the design of every new game level in SMM starts with an empty scenario, where the user is free to draw the level from scratch by picking game objects among hundreds of predefined options that lead to billions of possible object combinations. Therefore, it is very likely that most levels created will have one irrelevant similarity: the absence of many game objects. With that in mind, we propose to filter out clusters that have more than \( \lambda \) percent sparsity. That is, a cluster is removed if more than \( \lambda \) percent of its occurrence counts are zeros, only considering the features of the cluster’s subspace. For the same reason, we also prune clusters whose boundaries are \([0,0]\) in at least one feature of their corresponding subspaces. In our case study, \( \lambda = 50\% \) was used; it was chosen to be similar to the sparsity of the entire dataset. Once the aforementioned procedure was complete, only 318 clusters remained. They were further analyzed for additional pruning.
In the previous paragraph, we aimed at removing irrelevant clusters that were formed by the absence of game objects. Now, we focus on pruning irrelevant clusters based on their sizes; that is, based on the number of game levels that they contain. Figure 33 reports the rank of clusters sorted according to their sizes. As can be seen, there are a few large clusters that do not present much information because they include nearly all of the 720 game levels of our dataset. With that in mind, we propose to filter large clusters out of our analysis. Specifically, we pruned the 25 largest clusters. Note that their sizes vary from 718 to 583, while the 26th largest cluster has only 70 levels. For the sake of completeness, these clusters are detailed in Table 8.

**Table 8 – Top-25 clusters according to their sizes**

| Rank | Size  | Subspace |
|------|-------|----------|
| 1    | 718   | \(<F_{<C,(robust,stable,big)}>\,\{0,102\}\>> |
| 2    | 718   | \(<F_{<C,(teleport)}>\,\{0,150\}\>> |
| 3    | 717   | \(<F_{<C,(robust,stable)}>\,\{0,1557\}\>> |
| 4    | 715   | \(<F_{<C,(robust,stable,shooter)}>\,\{0,84\}\>> |
| 5    | 715   | \(<F_{<C,(support)}>\,\{0,773\}\>> |
| 6    | 713   | \(<F_{<C,(powerup)}>\,\{0,140\}\>> |
| 7    | 712   | \(<F_{<C,(bonus)}>\,\{0,595\}\>> |
| 8    | 702   | \(<F_{<C,(fragile,fly,big)}>\,\{0,9\}\>, \(<F_{<C,(robust,stable)}>\,\{0,1557\}\>> |
| 9    | 650   | \(<F_{<C,(vehicle)}>\,\{0,5\}, \,<F_{<C,(support)}>\,\{0,773\}\>> |
| 10   | 645   | \(<F_{<C,(support)}>\,\{0,191\}, \,<F_{<C,(bonus)}>\,\{0,595\}\>> |
| 11   | 643   | \(<F_{<C,(support)}>\,\{0,191\}, \,<F_{<C,(teleport)}>\,\{0,98\}\>> |
| 12   | 640   | \(<F_{<C,(robust,fly,shooter)}>\,\{0,2\}, \,<F_{<C,(support)}>\,\{0,191\}, \,<F_{<C,(bonus)}>\,\{0,595\}\>> |
| 13   | 640   | \(<F_{<C,(fragile,fly,shooter,big)}>\,\{0,1\}, \,<F_{<C,(support)}>\,\{0,191\}, \,<F_{<C,(bonus)}>\,\{0,582\}\>> |
| 14   | 638   | \(<F_{<C,(robust,fly,shooter)}>\,\{0,2\}, \,<F_{<C,(support)}>\,\{0,191\}, \,<F_{<C,(teleport)}>\,\{0,98\}\>> |
| 15   | 638   | \(<F_{<C,(fragile,fly,shooter,big)}>\,\{0,1\}, \,<F_{<C,(support)}>\,\{0,191\}, \,<F_{<C,(teleport)}>\,\{0,98\}\>> |
| 16   | 632   | \(<F_{<C,(fragile,fly,big)}>\,\{0,9\}, \,<F_{<C,(support)}>\,\{0,191\}, \,<F_{<C,(bonus)}>\,\{0,595\}\>> |
| 17   | 630   | \(<F_{<C,(fragile,fly,big)}>\,\{0,9\}, \,<F_{<C,(support)}>\,\{0,191\}, \,<F_{<C,(teleport)}>\,\{0,98\}\>> |
| 18   | 600   | \(<F_{<C,(robust,stable)}>\,\{0,199\}, \,<F_{<C,(vehicle)}>\,\{0,773\}\>> |
| 19   | 597   | \(<F_{<C,(robust,stable)}>\,\{0,199\}, \,<F_{<C,(vehicle)}>\,\{0,26\}\>> |
| 20   | 596   | \(<F_{<C,(robust,stable)}>\,\{0,199\}, \,<F_{<C,(fly,shooter)}>\,\{0,2\}, \,<F_{<C,(support)}>\,\{0,773\}\>> |
| 21   | 595   | \(<F_{<C,(robust,stable,big)}>\,\{0,17\}, \,<F_{<C,(support)}>\,\{0,773\}\>> |
| 22   | 595   | \(<F_{<C,(fragile,fly,shooter,big)}>\,\{0,1\}, \,<F_{<C,(robust,stable)}>\,\{0,199\}, \,<F_{<C,(support)}>\,\{0,773\}\>> |
| 23   | 592   | \(<F_{<C,(fragile,walk,shooter)}>\,\{0,5\}, \,<F_{<C,(robust,stable)}>\,\{0,199\}, \,<F_{<C,(support)}>\,\{0,773\}\>> |
| 24   | 585   | \(<F_{<C,(fragile,fly,big)}>\,\{0,9\}, \,<F_{<C,(robust,stable)}>\,\{0,199\}, \,<F_{<C,(support)}>\,\{0,773\}\>> |
| 25   | 583   | \(<F_{<C,(fragile,fly,big)}>\,\{0,9\}, \,<F_{<C,(robust,stable,big)}>\,\{0,17\}, \,<F_{<C,(support)}>\,\{0,773\}\>> |

Source: Research data.

Once the aforementioned procedure was complete, 293 clusters remained. Figure 34 reports the updated rank of clusters according to their sizes. Note that many tiny clusters also exist; they tend to be irrelevant due to their low representativity. With that in mind, we propose to use the rule of thumb 80/20 that comes from the Pareto principle (MANDELBROT, 1960) to perform one final round of pruning. The Pareto principle states that, for many events, roughly 80% of the effects come from 20% of the causes. It has been shown that many natural phenomena exhibit this property (NEWMAN, 2005). Due to this fact, the rule 80/20 is commonly used to select the most relevant examples, i.e., to “cut the long tail”, from data that exhibit skewed distributions, like the one shown in Figure 34. Therefore, we selected for our analysis the 59 largest clusters.
whose sizes go from 24 to 70 game levels; they refer to nearly 20% of the total 293 clusters. For the sake of completeness, Table 9 details the final set of clusters to be analyzed.

![Figure 33 – Head cut-off of clusters](image1)

![Figure 34 – Tail cut-off of clusters](image2)

Source: Elaborated by the author.

Source: Elaborated by the author.

| Category | Size | Subspace |
|----------|------|----------|
| C5       | 25   | \(<F \leq C, \{\text{fragile, fly, big}\} > 0.5\), \(<F \leq C, \{\text{fragile, fly, shooter, big}\} > 0.1\), \(<F \leq I, \{\text{vehicle}\} > 6.11\) > |
| C5       | 26   | \(<F \leq C, \{\text{robust, fly, shooter}\} > 0.2\), \(<F \leq I, \{\text{support}\} > 0.154\), \(<F \leq I, \{\text{bonus}\} > 0.250\) > |
| C5       | 26   | \(<F \leq C, \{\text{fragile, fly, shooter, big}\} > 0.1\), \(<F \leq C, \{\text{robust, fly, shooter}\} > 0.2\), \(<F \leq I, \{\text{vehicle}\} > 6.11\) > |
| C5       | 26   | \(<F \leq I, \{\text{support}\} > 0.154\), \(<F \leq I, \{\text{bonus}\} > 0.250\) > |
| C5       | 25   | \(<F \leq C, \{\text{fragile, fly, big}\} > 0.5\), \(<F \leq C, \{\text{fragile, fly, shooter, big}\} > 0.2\), \(<F \leq I, \{\text{vehicle}\} > 6.11\) > |
| C4       | 27   | \(<F \leq I, \{\text{bonus}\} > 0.250\) > |
| C6       | 38   | \(<F \leq I, \{\text{support}\} > 0.4\), \(<F \leq C, \{\text{fragile, walk, shooter, big}\} > 0.3\), \(<F \leq I, \{\text{vehicle}\} > 6.11\) > |
| C1       | 32   | \(<F \leq C, \{\text{robust, fly, shooter}\} > 0.2\), \(<F \leq I, \{\text{trip}\} > 0.399\) > |
| C1       | 31   | \(<F \leq C, \{\text{fragile, fly, big}\} > 0.5\), \(<F \leq C, \{\text{robust, stable, shooter}\} > 1.2536\) > |
| C2       | 30   | \(<F \leq C, \{\text{robust, stable, shooter}\} > 0.2536\), \(<F \leq I, \{\text{support}\} > 0.176\) > |
| C2       | 30   | \(<F \leq C, \{\text{fragile, fly, shooter, big}\} > 0.1\), \(<F \leq C, \{\text{fragile, fly, shooter, big}\} > 0.2\), \(<F \leq I, \{\text{vehicle}\} > 6.11\) > |
| C2       | 30   | \(<F \leq I, \{\text{support}\} > 0.154\) > |
| C2       | 29   | \(<F \leq C, \{\text{fragile, fly, shooter, big}\} > 0.6\), \(<F \leq C, \{\text{fragile, fly, shooter, big}\} > 0.1\), \(<F \leq I, \{\text{vehicle}\} > 6.11\) > |
| C2       | 29   | \(<F \leq I, \{\text{support}\} > 0.154\) > |
| C2       | 28   | \(<F \leq C, \{\text{fragile, fly, shooter, big}\} > 0.5\), \(<F \leq C, \{\text{fragile, fly, shooter, big}\} > 0.098\) > |
| C2       | 28   | \(<F \leq I, \{\text{vehicle}\} > 6.11\) > |

Continued on next page
### 5.3. Characterizing Popular Games

#### Table 9 – Final set of clusters selected for analysis (continued)

| Category | Size | Subspace |
|----------|------|----------|
| C4       | 28   | \(<F_{<C}.\{\text{fragile, fly, shooter, big}\}>[0.1]\), \(<F_{<C}.\{\text{robust, fly, shooter}\}>[0.2]\), \(<F_{<I}.\{\text{vehicle}\}>[16,11]\), \(<F_{<I}.\{\text{bonus}\}>[0.25]\) |
| C1       | 47   | \(<F_{<C}.\{\text{fragile, fly, big}\}>[0.9]\), \(<F_{<C}.\{\text{robust, vehicle, big}\}>[0.2]\) |
| C1       | 38   | \(<F_{<C}.\{\text{robust, stable, shooter}\}>[13,24]\), \(<F_{<I}.\{\text{support}\}>[0.398]\) |
| C1       | 59   | \(<F_{<C}.\{\text{robust, stable, big}\}>[0.5]\), \(<F_{<C}.\{\text{robust, walk, shooter, big}\}>[0.5]\), \(<F_{<C}.\{\text{vehicle}\}>[0.5]\), \(<F_{<I}.\{\text{support}\}>[0.154]\) |
| C4       | 62   | \(<F_{<C}.\{\text{robust, stable, big}\}>[0.5]\), \(<F_{<C}.\{\text{vehicle}\}>[18,34]\), \(<F_{<I}.\{\text{support}\}>[0.176]\) |
| C3       | 24   | \(<F_{<C}.\{\text{robust, stable, big}\}>[0.5]\), \(<F_{<C}.\{\text{robust, stable, big}\}>[0.11]\) |
| C4       | 63   | \(<F_{<C}.\{\text{robust, stable, big}\}>[0.5]\), \(<F_{<C}.\{\text{vehicle}\}>[18,34]\), \(<F_{<I}.\{\text{support}\}>[0.176]\) |
| C4       | 28   | \(<F_{<C}.\{\text{robust, fly, shooter}\}>[0.2]\), \(<F_{<I}.\{\text{vehicle}\}>[6,11]\), \(<F_{<I}.\{\text{bonus}\}>[0.25]\) |
| C4       | 28   | \(<F_{<C}.\{\text{robust, walk, shooter, big}\}>[0.5]\), \(<F_{<C}.\{\text{vehicle}\}>[6,11]\), \(<F_{<I}.\{\text{bonus}\}>[0.25]\) |
| C4       | 63   | \(<F_{<C}.\{\text{vehicle}\}>[0.11]\) |
| C1       | 63   | \(<F_{<C}.\{\text{vehicle}\}>[0.11]\) |
| C1       | 29   | \(<F_{<C}.\{\text{robust, stable, big}\}>[0.5]\), \(<F_{<C}.\{\text{vehicle}\}>[6,11]\), \(<F_{<I}.\{\text{bonus}\}>[0.25]\) |
| C2       | 29   | \(<F_{<C}.\{\text{robust, walk, shooter, big}\}>[0.5]\), \(<F_{<C}.\{\text{vehicle}\}>[6,11]\), \(<F_{<I}.\{\text{support}\}>[0.098]\) |
| C3       | 30   | \(<F_{<C}.\{\text{robust, stable, shooter}\}>[25,36]\), \(<F_{<C}.\{\text{vehicle}\}>[6,11]\), \(<F_{<I}.\{\text{vehicle}\}>[0.2]\) |
| C1       | 49   | \(<F_{<C}.\{\text{vehicle}\}>[0.11]\) |
| C1       | 59   | \(<F_{<C}.\{\text{robust, stable, big}\}>[0.5]\), \(<F_{<C}.\{\text{robust, stable, big}\}>[0.398]\) |
| C3       | 58   | \(<F_{<C}.\{\text{robust, walk, shooter, big}\}>[0.5]\), \(<F_{<C}.\{\text{vehicle}\}>[201,392]\), \(<F_{<I}.\{\text{vehicle}\}>[0.11]\) |
| C3       | 24   | \(<F_{<C}.\{\text{robust, stable, big}\}>[0.5]\), \(<F_{<C}.\{\text{vehicle}\}>[35,51]\), \(<F_{<I}.\{\text{price}\}>[0.3]\) |
| C2       | 31   | \(<F_{<C}.\{\text{robust, stable, big}\}>[0.5]\), \(<F_{<C}.\{\text{vehicle}\}>[6,11]\), \(<F_{<I}.\{\text{support}\}>[0.154]\) |
| C1       | 60   | \(<F_{<C}.\{\text{robust, stable, big}\}>[0.5]\), \(<F_{<C}.\{\text{vehicle}\}>[201,392]\), \(<F_{<I}.\{\text{support}\}>[0.299]\) |
| C3       | 59   | \(<F_{<C}.\{\text{vehicle}\}>[0.11]\) |
| C1       | 32   | \(<F_{<C}.\{\text{robust, stable, shooter}\}>[25,36]\), \(<F_{<C}.\{\text{vehicle}\}>[6,11]\), \(<F_{<I}.\{\text{vehicle}\}>[0.11]\) |
| C1       | 59   | \(<F_{<C}.\{\text{robust, stable, big}\}>[0.5]\), \(<F_{<C}.\{\text{robust, stable, big}\}>[0.299]\) |
| C6       | 38   | \(<F_{<C}.\{\text{robust, stable, big}\}>[0.4]\), \(<F_{<C}.\{\text{robust, stable, big}\}>[0.399]\), \(<F_{<I}.\{\text{vehicle}\}>[0.399]\) |
| C1       | 39   | \(<F_{<C}.\{\text{robust, stable, big}\}>[0.3]\), \(<F_{<I}.\{\text{vehicle}\}>[0.399]\), \(<F_{<I}.\{\text{vehicle}\}>[0.03]\) |
| C1       | 25   | \(<F_{<C}.\{\text{vehicle}\}>[0.11]\) |
| C2       | 30   | \(<F_{<C}.\{\text{robust, fly, shooter}\}>[0.2]\), \(<F_{<C}.\{\text{vehicle}\}>[6,11]\), \(<F_{<I}.\{\text{vehicle}\}>[0.154]\) |
| C1       | 25   | \(<F_{<C}.\{\text{robust, stable, big}\}>[0.3]\), \(<F_{<C}.\{\text{vehicle}\}>[0.398]\) |
| C3       | 25   | \(<F_{<C}.\{\text{robust, stable, big}\}>[0.3]\), \(<F_{<C}.\{\text{vehicle}\}>[0.03]\) |
| C7       | 70   | \(<F_{<C}.\{\text{robust, stable, big}\}>[18,34]\), \(<F_{<C}.\{\text{vehicle}\}>[0.2]\) |

*Continued on next page*
Table 9 – Final set of clusters selected for analysis (continued)

| Category | Size | Subspace |
|----------|------|----------|
| C3       | 30   | \(<F_{<C,\{\text{robust,stable,shooter}\}>[25,36]}, F_{<I,\{\text{vehicle}\}>[0,11]}>)\) |
| C3       | 67   | \(<F_{<C,\{\text{robust,stable,big}\}>[18,34]}, F_{<I,\{\text{vehicle}\}>[0,11]}>)\) |
| C7       | 34   | \(<F_{<C,\{\text{robust,stable,shooter}\}>[25,36]}, F_{<C,\{\text{robust,fly,shooter}\}>[0,2]}>)\) |
| C1       | 40   | \(<F_{<I,\{\text{support}\}>[203,399]}, F_{<I,\{\text{bonus}\}>[0,296]}>)\) |
| C6       | 42   | \(<F_{<I,\{\text{support}\}>[203,399]}, F_{<I,\{\text{teleport}\}>[0,58]}>)\) |
| C3       | 52   | \(<F_{<C,\{\text{robust,stable,shooter}\}>[13,24]}, F_{<I,\{\text{vehicle}\}>[0,13]}>)\) |
| C3       | 59   | \(<F_{<C,\{\text{robust,stable}\}>[201,392]}, F_{<I,\{\text{vehicle}\}>[0,11]}>)\) |
| C1       | 60   | \(<F_{<C,\{\text{robust,stable}\}>[201,392]}, F_{<I,\{\text{support}\}>[0,299]}>)\) |
| C1       | 64   | \(<F_{<C,\{\text{robust,stable,big}\}>[18,34]}, F_{<I,\{\text{support}\}>[0,176]}>)\) |
| C2       | 31   | \(<F_{<I,\{\text{vehicle}\}>[6,11]}, F_{<I,\{\text{support}\}>[0,154]}>)\) |

Source: Research data.

5.3.1.3 Analyzing Clusters of Similar Game Levels

At this point, there are selected 59 clusters of popular levels from a game of platform that is well-known worldwide. Each level is represented by the histogram of categorized game objects that it contains. Therefore, the clusters describe combinations of items and characters that frequently occur in the 720 popular levels of our case study. These clusters are analyzed in the following with focus on understanding and reporting their semantics.

Each cluster was studied individually by observing the features that form its subspace and the corresponding maximum and minimum numbers of objects that define the cluster’s boundaries. This process led us to realize that some clusters share similar properties. Note that it was already expected because algorithm CLIQUE tends to infer many redundant clusters (MOISE; SANDER; ESTER, 2006); a cluster in a subspace with \(d\) features always has a counterpart in each subspace of lower dimensionality formed by one subset of these features, and CLIQUE reports them all. Provided that redundant clusters were identified, we manually distributed them into seven categories; see the first column of Table 9 for details. The categories represent sets of clusters that share similar subspaces and boundaries. They are described in the following.

C1 – This category has the largest number of clusters, 23. In general, it represents game levels that contain support and bonus items together with several opponents. This is a common game combination, since most games of platform have many opponents and items of these kinds. It is also possible to note large intervals defining the clusters’ boundaries. For example, feature \(F_{<I,\{\text{support}\}>}\) ranges from 0 to 398. Thus, there are no apparent hints on how many items and opponents should be used when designing a new game level. To the best of our knowledge, C1 does not present any valid or new knowledge, beyond the obvious;
5.3. Characterizing Popular Games

C2 – This category has 11 clusters with nearly 39 game levels each. It presents combinations of several opponents, vehicle-related items and items of support. Once more, there is a strong presence of support items with feature $F_{<I,\{support\}}$ ranging from 0 to 120, approximately. It may be possible to identify useful patterns from C2. However, the same patterns are more clear and better detailed in other categories, so we elaborate on them in the following;

C3 – This category has 10 clusters. It mainly presents combinations of opponents and vehicle-related items. The most interesting ones are described in the following:

(1) – Many clusters have features $F_{<C,\{robust,stable,shooter\}}$ and $F_{<I,\{vehicle\}}$ in their subspaces. There are game levels with an elevated number of vehicles and a few opponents that are robust, stable and have the ability to shoot. It seems to be a successful combination that generates popular games with a normal degree of difficulty. Interestingly, the opposite combination is also common; that is, some games have very few vehicles and many opponents with the aforementioned characteristics. This combination is also successful, and it seems to generate games that are more difficult;

(2) – Two other clusters are formed in the subspace of features $F_{<C,\{robust,stable,big\}}$ and $F_{<I,\{vehicle\}}$. One cluster has boundaries at [35, 51] and [0, 3] for these features, respectively. It probably includes difficult game levels because they have many large opponents that are robust and stable, while there are very few vehicles. The other cluster has boundaries at [18, 34] and [0, 11] for the same features, respectively. Thus, it may include easier games. In both cases, successful game levels are created.

(3) – A successful combination to generate popular games is also described by clusters in the subspace of features $F_{<C,\{fragile,fly,big\}}$, $F_{<C,\{robust,stable,big\}}$ and $F_{<I,\{vehicle\}}$. Apparently, this combination tends to generate difficult games because they include many large opponents that are robust and stable, a moderate number of large opponents that are fragile and can fly, and very few vehicles;

(4) – There is another successful combination that refers to the use of vehicle-related items together with specific types of opponents. It is explained by a cluster whose boundaries are [25, 36], [0, 2] and [0, 11] with regard to features $F_{<C,\{robust,stable,shooter\}}$, $F_{<C,\{robust,fly,shooter\}}$ and $F_{<I,\{vehicle\}}$, respectively;

(5) – Finally, it is worth noting that the subspaces of three other clusters include feature $F_{<C,\{robust,stable\}}$ with boundaries [201, 392]. It seems to be one successful combination, this time referring to very difficult games that have many opponents;

C4 – This category has 6 clusters with an average size of nearly 28 game levels each. It presents interesting combinations of specific types of opponents, vehicles and bonus items. One may consider it obvious that many game levels have bonus items. However, in this category we always find opponents that have the ability to fly in game levels whose counts of vehicles are between 6 and 11, and the counts of bonus items are between
0 and 250. These opponents are the ones represented by features $F_{<C,\{\text{fragile, fly, big}\}}$, $F_{<C,\{\text{fragile, fly, shooter, big}\}}$ and $F_{<C,\{\text{robust, fly, shooter}\}}$. Therefore, it seems that these 3 types of flying opponents present one useful combination for game levels that have many bonus items and a few vehicles.

**C5** – This category has 4 clusters with an average size of nearly 25 game levels each. It presents combinations of support and bonus items. The clusters occur in the subspace of features $F_{<I,\{\text{support}\}}$ and $F_{<I,\{\text{bonus}\}}$ with boundaries at $[0, 154]$ and $[0, 250]$, respectively. Due to the very large intervals, this category does not present any specific and valid knowledge.

**C6** – This category has 3 clusters with an average size of nearly 40 levels each. It presents combinations of support items and items of teleportation that occur together with some opponents. All clusters exist in subspaces that include features $F_{<I,\{\text{support}\}}$ and $F_{<I,\{\text{teleport}\}}$ with boundaries fixed at $[206, 399]$ and $[0, 39]$, respectively. Due to the large intervals, this combination does not seem to present any specific information by itself, even when considering the occurrence of opponents. So, we could not spot any specific and valid knowledge here.

**C7** – This is the only category that exclusively presents combinations of opponents. It has two clusters; one with 70 game levels and the other with 34 levels. Both clusters have 2-dimensional subspaces including feature $F_{<C,\{\text{robust, fly, shooter}\}}$ with boundaries at $[0, 2]$. Therefore, they refer to game levels with very few robust opponents that have the abilities to fly and shoot. However, the clusters differ on the other feature of their subspaces; one of them includes feature $F_{<C,\{\text{robust, stable, big}\}}$ with boundaries at $[18, 34]$, while the other has feature $F_{<C,\{\text{robust, stable, shooter}\}}$ with boundaries at $[25, 36]$. They both refer to robust and stable opponents, though. Interestingly, these clusters’ boundaries are defined by very small intervals, so they may provide valuable hints on how many opponents of the aforementioned types should be used when designing a new game level. Also, those opponents have appeared on rare occasions in the previous categories. Therefore, this category seems to present useful combinations that are built upon those types of opponents.

In summary, this section reported and discussed many combinations of game objects that frequently occur in 720 successful levels selected from a very popular games of platform. We believe that this knowledge may be useful for designers when elaborating new games. It was made possible by following a threefold procedure: (i) to identify successful levels that were not created by game influencers, so to avoid bias in the analysis; (ii) to represent each game level as the histogram of categorized objects that it contains, and; (iii) to take advantage of subspace clustering to spot the most frequent combinations of items and characters, also using a novel heuristic to ignore irrelevant clusters. Each remaining cluster describes a successful combination of game objects. Therefore, we manually evaluated the clusters and grouped them into seven categories that have unique characteristics. Let us emphasize that the most interesting patterns
were found into 18 clusters from C3, C4 and C7. The next section evaluates the interdependence of these object combinations with the discovery and the analysis of association rules.

5.3.2 Analyzing Popular Games with Association Rules

At this point, there are already analyzed 59 clusters of successful game levels aimed at the discovery of useful and valid knowledge from frequent combinations of items and opponents. Many clusters presented patterns that are obvious or unspecific. Nevertheless, very interesting object combinations were detected into 18 clusters from C3, C4 and C7. In this section, we further investigate these combinations by taking advantage of association rules.

As described in Chapter 2, association rules are frequent co-occurrences of specific values on subsets of features. In our context, they are well-suited to describe patterns of interdependence regarding game items and opponents. For example, one association rule can indicate that game levels with vehicles have high probability to also include opponents that are robust and stable. With that in mind, we propose to take advantage of these rules to quantify and evaluate the interdependence that may exist in the occurrence of the game objects that belong to the combinations considered to be interesting in the previous section. To make it possible, we use the well-known metrics in the following: support, confidence, lift, conviction and leverage. Figure 35 illustrates the procedure that we propose. At first, frequent itemsets are identified among the interesting combinations. Then, they are used to detect and evaluate association rules.

Figure 35 – Analyzing popular games with association rules

(1) Frequent itemsets

\{ F_{I,<\text{vehicle}>} \}, \{ F_{C,<\text{robust,stable}>} \}
\{ F_{I,<\text{vehicle}>} \}, \{ F_{C,<\text{fragile,fly,big}>} \}
\{ F_{I,<\text{vehicle}>} \}, \{ F_{C,<\text{robust,stable}>} \}
...

(2) Rules

\{ F_{I,<\text{vehicle}>} \} \rightarrow \{ F_{C,<\text{robust,stable}>} \}
\{ F_{I,<\text{vehicle}>} \} \rightarrow \{ F_{C,<\text{fragile,fly,big}>} \}
\{ F_{I,<\text{vehicle}>} \} \rightarrow \{ F_{C,<\text{robust,stable}>} \} \rightarrow \{ F_{C,<\text{fragile,fly,big}>} \}
...

Source: Elaborated by the author.

5.3.2.1 Discovering Association Rules

At first, we preprocessed our dataset of categorized object counts in order to obtain a binary matrix. Instead of spotting rules from exact object counts, we propose to analyze a matrix that simply differentiates the presence and the absence of each type of object into each game level. Therefore, non-zero occurrence counts were transformed into ones, while the zero counts remained untouched. We believe that this procedure may allow the discovery of more rules.

In the sequence, we took advantage of the algorithm Apriori (AGRAWAL; SRIKANT et al., 1994) in order to identify itemsets that frequently occur into the binary matrix. Apriori was configured to return itemsets with support equal to or greater than 0.05. Note that, in our case
study, a frequent itemset refers to one combination of game objects that occurs in many game levels. Provided that there are 720 levels, the aforementioned procedure allowed us to identify every combination of objects that appears in at least 36 distinct levels. In total, 93,825 combinations were identified. Among them, only seven combinations were also considered to be interesting in the clustering-based analysis. They are reported in Table 10.

Table 10 – Frequent itemsets sorted by their support

| Identifier | Support | Itemset |
|------------|---------|---------|
| I₁         | 0.3417  | \( F_{<C, \{\text{robust, stable}\}} \cdot F_{<I, \{\text{vehicle}\}} \) |
| I₂         | 0.2681  | \( F_{<C, \{\text{robust, stable, big}\}} \cdot F_{<I, \{\text{vehicle}\}} \) |
| I₃         | 0.2681  | \( F_{<C, \{\text{robust, stable, shooter}\}} \cdot F_{<I, \{\text{vehicle}\}} \) |
| I₄         | 0.0778  | \( F_{<C, \{\text{robust, stable}\}} \cdot F_{<C, \{\text{fragile, fly, big}\}} \cdot F_{<I, \{\text{vehicle}\}} \) |
| I₅         | 0.0708  | \( F_{<C, \{\text{robust, stable, big}\}} \cdot F_{<C, \{\text{fragile, fly, big}\}} \cdot F_{<I, \{\text{vehicle}\}} \) |
| I₆         | 0.0708  | \( F_{<I, \{\text{bonus}\}} \cdot F_{<C, \{\text{fragile, fly,big}\}} \cdot F_{<I, \{\text{vehicle\}} \} \) |
| I₇         | 0.0583  | \( F_{<C, \{\text{fragile, walk, shooter, big\}} \cdot F_{<C, \{\text{robust, stable\}} \cdot F_{<I, \{\text{vehicle\}} \} \} \) |

Source: Research data.

The itemsets of Table 10 were then used to spot association rules with confidence equal to or greater than 0.5. For a rule \( X \rightarrow Y \), confidence is the probability that the consequent \( Y \) will happen given the occurrence of the antecedent \( X \). Thus, we consider rules in which \( Y \) happens in at least half of the occurrences of \( X \). In total, 20 rules were found. They are reported in Table 11.

5.3.2.2 Analyzing Association Rules

This section evaluates the rules of Table 11. In addition to confidence, we consider the well-known metrics lift, conviction and leverage. All rules present values of lift, conviction and leverage that guarantee some degree of dependence between each consequent and its antecedent. That is, the occurrence of the consequent depends on whether or not the antecedent occurs, in each case with its own level of reliability. For metrics lift and conviction, values higher than one mean dependence; leverage values greater than zero have the same meaning.

Rule \( R₁ \) presents 1.0 confidence, thus the consequent \( \{ F_{<C, \{\text{robust, stable}\}} \} \) strongly depends on the antecedent \( \{ F_{<C, \{\text{fragile, walk, shooter, big\}} \cdot F_{<I, \{\text{vehicle}\}} \} \). However, this rule has only 0.06 of support, so it rarely occurs. As discussed in Category C3 (5) from Section 5.3.1.3, this combination appears in clusters of game levels that seem to have a very high degree of difficulty. The levels include hundreds of opponents that are robust and stable, very few vehicles and a few opponents of type \( F_{<C, \{\text{fragile, walk, shooter, big\}} \). Meanwhile, as can be seen in Rule \( R₂ \), the support grows to 0.11 when \( F_{<I, \{\text{vehicle}\}} \) is removed from the antecedent, while the confidence remains very high, \( i.e., \) 0.98. This fact reinforces that combining opponents of types \( F_{<C, \{\text{robust, stable}\}} \) and \( F_{<C, \{\text{fragile, walk, shooter, big\}} \) may be a successful option.

Rule \( R₃ \) is similar to Rule \( R₁ \). It also presents a high confidence and a low support, besides having the same consequent \( \{ F_{<C, \{\text{robust, stable}\}} \} \) and an antecedent with vehicle-related items.
5.3. Characterizing Popular Games

Table 11 – Association rules sorted by their confidence

| Identifier | Antecedent X | Consequent Y | SUPP | CONF | LIFT | CONV | LEV |
|------------|--------------|--------------|------|------|------|------|-----|
| R1 | $\{F_{<C, \{fragile, walk, shooter, big\}}\}$, $F_{<1, \{vehicle\}}$ | $\{F_{<C, \{robust, stable\}}\}$ | 0.06 | 1.00 | 1.14 | $\infty$ | 0.01 |
| R2 | $\{F_{<C, \{fragile, walk, shooter, big\}}\}$ | $\{F_{<C, \{robust, stable\}}\}$ | 0.11 | 0.98 | 1.11 | 5.07 | 0.01 |
| R3 | $\{F_{<C, \{fragile, fly, big\}}\}$, $F_{<1, \{vehicle\}}$ | $\{F_{<C, \{robust, stable\}}\}$ | 0.08 | 0.97 | 1.10 | 3.54 | 0.01 |
| R4 | $F_{<1, \{vehicle\}}$ | $\{F_{<C, \{robust, stable\}}\}$ | 0.34 | 0.93 | 1.06 | 1.70 | 0.02 |
| R5 | $\{F_{<C, \{fragile, fly, big\}}\}$ | $\{F_{<C, \{robust, stable\}}\}$ | 0.14 | 0.92 | 1.04 | 1.47 | 0.01 |
| R6 | $\{F_{<C, \{fragile, fly, big\}}\}$, $F_{<1, \{vehicle\}}$ | $\{F_{<C, \{robust, stable\}}\}$ | 0.07 | 0.88 | 1.13 | 1.84 | 0.01 |
| R7 | $\{F_{<C, \{fragile, fly, big\}}\}$, $\{F_{<C, \{robust, stable, big\}}\}$, $F_{<1, \{vehicle\}}$ | $\{F_{<C, \{robust, stable, big\}}\}$ | 0.07 | 0.88 | 1.39 | 3.06 | 0.02 |
| R8 | $\{F_{<1, \{vehicle\}}\}$ | $\{F_{<1, \{bonus\}}\}$ | 0.32 | 0.86 | 1.10 | 1.55 | 0.03 |
| R9 | $\{F_{<C, \{fragile, fly, big\}}\}$ | $\{F_{<1, \{bonus\}}\}$ | 0.13 | 0.85 | 1.10 | 1.50 | 0.01 |
| R10 | $\{F_{<C, \{fragile, fly, big\}}\}$ | $\{F_{<C, \{robust, stable, big\}}\}$ | 0.12 | 0.78 | 1.23 | 1.66 | 0.02 |
| R11 | $F_{<1, \{vehicle\}}$ | $\{F_{<C, \{robust, stable, big\}}\}$ | 0.27 | 0.73 | 1.16 | 1.36 | 0.04 |
| R12 | $\{F_{<C, \{robust, stable, big\}}, F_{<1, \{vehicle\}}\}$ | $\{F_{<C, \{robust, stable, shooter\}}\}$ | 0.27 | 0.73 | 1.23 | 1.49 | 0.05 |
| R13 | $\{F_{<C, \{robust, stable, big\}}, F_{<1, \{vehicle\}}\}$ | $\{F_{<C, \{robust, stable, big\}}\}$ | 0.07 | 0.61 | 1.65 | 1.61 | 0.03 |
| R14 | $\{F_{<C, \{robust, stable, big\}}, F_{<C, \{fragile, fly, big\}}\}$ | $\{F_{<1, \{vehicle\}}\}$ | 0.08 | 0.57 | 1.54 | 1.45 | 0.03 |
| R15 | $\{F_{<1, \{bonus\}}, F_{<C, \{fragile, fly, big\}}\}$ | $\{F_{<C, \{robust, stable, big\}}\}$ | 0.07 | 0.55 | 1.51 | 1.42 | 0.02 |
| R16 | $\{F_{<C, \{fragile, fly, big\}}\}$ | $\{F_{<1, \{vehicle\}}\}$ | 0.08 | 0.54 | 1.46 | 1.37 | 0.03 |
| R17 | $\{F_{<C, \{fragile, fly, big\}}\}$ | $\{F_{<C, \{robust, stable\}}\}$ | 0.08 | 0.52 | 1.52 | 1.37 | 0.03 |
| R18 | $\{F_{<C, \{fragile, walk, shooter, big\}}, F_{<C, \{robust, stable\}}\}$ | $\{F_{<C, \{robust, stable, big\}}\}$ | 0.06 | 0.52 | 1.41 | 1.31 | 0.02 |
| R19 | $\{F_{<C, \{fragile, walk, shooter, big\}}\}$ | $\{F_{<C, \{robust, stable, big\}}\}$ | 0.06 | 0.51 | 1.48 | 1.33 | 0.02 |
| R20 | $\{F_{<C, \{fragile, walk, shooter, big\}}\}$ | $\{F_{<1, \{vehicle\}}\}$ | 0.06 | 0.51 | 1.37 | 1.28 | 0.02 |

Source: Research data.

The distinction is that the antecedent includes opponents of type $F_{<C, \{fragile, fly, big\}}$ instead of those of type $F_{<C, \{fragile, walk, shooter, big\}}$. Thus, Rule R3 defines another successful combination involving vehicles and opponents that are robust and stable. Meanwhile, Rule R5 can be seen as a decomposition of Rule R3, where $\{F_{<C, \{fragile, fly, big\}}\}$ induces $\{F_{<C, \{robust, stable\}}\}$ with support 0.14 and confidence 0.92. Finally, Rule R4 shows that $\{F_{<1, \{vehicle\}}\}$ also induces $\{F_{<C, \{robust, stable\}}\}$. This rule has a relatively high support, i.e., 0.34, and the confidence is 0.93.

Besides the aforementioned cases, the combination of some specific opponents and vehicle-related items induces the occurrence of opponents of other types. In Rule R7, the antecedent $\{F_{<1, \{vehicle\}}, F_{<C, \{fragile, fly, big\}}\}$ induces the consequent $\{F_{<C, \{robust, stable, big\}}\}$ with support 0.07 and confidence 0.88. As discussed in Category C3 (3) from Section 5.3.1.3, this combination can be used to elaborate either very difficult game levels or easy ones. Also, Rule R10 presents a decomposition of Rule R7, in which the antecedent $\{F_{<C, \{fragile, fly, big\}}\}$ induces the consequent $\{F_{<C, \{robust, stable, big\}}\}$ with support 0.12 and confidence 0.78. Note that opponents of these types do not appear alone in the clusters described in Section 5.3.1.3; they always occur together with vehicles, but, still, it seems to exist a strong correlation between the occurrence of these opponents combined in one game level and the success of the level.
Rules $R_{11}$ and $R_{12}$ are conditioned by the occurrence of vehicles; that is, their antecedent is $\{F_{<1,\{\text{vehicle}\}}>\}$. Depending on the rule, this antecedent induces $\{F_{<C,\{\text{robust,stable,big}\}}>\}$ or $\{F_{<C,\{\text{robust,stable,shooter}\}}>\}$, but in both cases the support is 0.27 and the confidence is 0.73. Thus, the presence of vehicles in successful levels may condition the occurrence of some specific opponents, as well as of the combinations thereof that were discussed in the previous paragraphs.

Finally, Rules $R_6$, $R_8$ and $R_9$ have $\{F_{<1,\{\text{bonus}\}}>\}$ as the consequent. We consider that these rules present obvious combinations since nearly all game levels of our case study have bonus items, as happens in potentially any game of platform. Also, Rules $R_{12}$ to $R_{20}$ constitute minor variations of the rules described in the previous paragraphs. They reinforce that the combinations mentioned are strongly conditioned, having always confidence values above 0.5.

5.4 Discussion

When elaborating a new game of platform, the designers commonly have to select characters and items from a set of predefined options that lead to thousands of possible combinations. In this scenario, how to identify those combinations that have an elevated potential of success, so to support the work of non-expert game designers? To answer the question, this chapter presented one case study on the automatic detection of non-obvious combinations of game characters and items that commonly occur in popular levels from the well-known game of platform Super Mario Maker. It is the third and last main contribution of this MSc research.

At first, we decided to avoid bias in the analysis by filtering out of our study popular levels that were produced by influencer developers. Therefore, we identified game influencers in our dataset using the approach described in Chapter 4, and then removed from our analysis the levels that they produced. Next, we proposed novel approaches for feature extraction and feature aggregation aimed at representing each game level as the histogram of categorized game objects that it contains. The feature aggregation step was designed with focus in obtaining general game characteristics, so to make our proposal potentially suitable for any game of platform, not only for Super Mario Maker. Inspired by the work of Beaupre et al. (2018), the aggregation was performed by following one novel hierarchy of object types and subtypes that represents a more realistic scenario of the design of games. Note that our current hierarchy explores only interactable items and harmful opponents, but this strategy can also be used to consider other types of game objects as well. Finally, we performed a series of analyses in order to discover the most successful object combinations: (1) subspace clustering to detect groups of game levels that share similar object counts with regard to specific subsets of object categories; (2) automatic and manual processes to prune irrelevant clusters; (3) extensive cluster analysis to provide insights into the most successful object combinations; and, (4) a complementary study based on association rules to evaluate dependencies that exist in the occurrence of objects.

Thus, with this case study, we aim to support game designers in the elaboration of new,
successful games of platform by taking advantage of the object combinations that we identified as well as the feature aggregation that we proposed. Also, we hope that researchers and designers will find creative, additional uses for our results. The next chapter concludes this MSc research.
This chapter presents the conclusion and highlights the main contributions of the MSc research. It contains brief descriptions on the results, their impact and future work.

6.1 Main Contributions

Techniques of Data Mining have been recently applied in the field of digital games both to extract knowledge from the games themselves and from their players. This MSc research explored such techniques into the context of Social Networks of Games. We were particularly interested in the discovery of relevant characteristics that are common to popular games of platform. The well-known game Super Mario Maker was used to make it possible by means of a case study. Here, it is important to highlight that we avoided bias in our results by only analyzing games that were developed by non-influencer users, which was required because game influencers may attract attention due to their popularity, and not necessarily due to the quality of their work. In this sense, our research was motivated by the following hypothesis:

*It is possible to use information from a Social Network of Games to support the development of new, more successful games by taking advantage of Knowledge Discovery in Databases to identify common characteristics present in popular games that were elaborated by non-influencer makers, so to avoid bias in the process.*

As a result, we coined the term Social Network of Games, and we also crawled a dataset based on a real network that is well-known worldwide. A novel framework to detect game influencers was also developed. Finally, we performed an extensive analysis on relevant features of popular games and identified general object combinations that were commonly used with success in previous games, so to support the development of new games of platform that are even more attractive. With that in mind, we argue that the aforementioned hypothesis was corroborated by our work. In summary, this Msc research produced three main contributions:
C1. **Social Network of Games** – in Chapter 3, we coined the term Social Network of Games and presented a formal definition for it. We also described the data collection process that was performed for the well-known game Super Mario Maker, and reported statistics on a dataset generated by this process. To the best of our knowledge, it is the *first* public dataset collected from a real Social Network of Games. This dataset describes nearly seven million interactions that were performed by more than 880 thousand players on over 115 thousand game levels. We also highlighted the vast applicability of this data in the research areas of Social Network Analysis and Artificial Intelligence in Games. Moreover, our dataset generated the following publication, (MORAES; CORDEIRO, 2019b);

C2. **Detection of Game Influencers** – Chapter 4 presented a novel framework to detect game influencers in Social Networks of Games. It includes the feature extractors Linear Regression, Delta Rank and Coefficient of Angle. They are all unsupervised and based on temporal aspects of the players’ actions within the network. In the experimental evaluation, 28 classification algorithms were studied; the best classifier obtained an accuracy of 87.1% and a F1-score of 85.7% by taking advantage of our features. We also demonstrated that our proposal is generic enough to accurately model the behavior of game influencers that come from different countries. Additionally, our framework can in theory be used by non-game-related applications to spot influencers regarding other social network scenarios. Moreover, this work generated the following publication, (MORAES; CORDEIRO, 2019a);

C3. **Characterization of Popular Games** – in Chapter 5, we presented a case study aimed at identifying combinations of game objects that commonly occur within popular game levels. To make it possible, it was proposed a novel design pattern that categorizes game objects into types and subtypes, so to make our proposals general enough to support the design of potentially any game of platform. Next, we performed a series of analyses using subspace clustering and association rules to identify successful and non-obvious object combinations. Finally, we discussed the main combinations identified and their applicability in the hope of being helpful to improve the ability of non-expert game designers to elaborate new games that are even more interesting. Moreover, this particular contribution is included in one manuscript that is currently being written for a journal submission.

### 6.2 Future Work

Each of the aforementioned contributions enables future work. The main possibilities are described in the following:

**C1.** The dataset presented in Chapter 3 has a vast applicability in the area of Social Network Analysis. It can be used for: (1) community detection, such as to identify communities of similar players; (2) link prediction, for example, to predict what games a given player
will play or like, and; (3) ranking, e.g., to sort games by their numbers of likes. Our dataset also paves the way for applications in Artificial Intelligence. For example, data mining techniques may predict the popularity of one newly developed game, while player modeling strategies may help to infer the preferences of players based on their activities;

C2. The detector of influencers presented in Chapter 4 was developed for Social Networks of Games. However, it can in theory be used for non-game-related applications. As further research, one can analyze other social networks to spot digital influencers. Also, our proposed feature extractors may be helpful in regression analyses, so to study the different degrees of influence that influencers may have within game-related networks or others;

C3. The case study presented in Chapter 5 can be further improved by exploring object occurrences within specific locations, which may be interesting since players usually interact with objects that are nearby in a game level. Therefore, one may take into account the position of objects within a level in a future work, instead of only counting their occurrences as done in this MSc research. Additionally, our feature aggregation strategy can be further improved by considering non-interactable items and characters.

6.3 Additional Contribution

Besides the aforementioned contributions, we produced one additional work in collaboration with other researchers from the Database and Image Group (GBdI) at the Institute of Mathematics and Computer Science (ICMC) of the University of São Paulo (USP). In Kunze, et al. (2018), we explored temporal aspects of time series containing measurements of Normalized Difference Vegetation Index (NDVI) that were extracted from remote sensing images in order to identify sugarcane crops following a classification approach.

6.4 Awards

Finally, two awards were received as a consequence of this MSc research:

- The API developed to retrieve data from player profiles in the website SMM Bookmark received the Innovation Award from JavaScript Classes in August 2017;
- The publication in Moraes and Cordeiro (2019a) was selected as one of the best papers of the 21st International Conference on Enterprise Information Systems (ICEIS 2019). We were then invited to extend the work to be considered for publication at the Springer Lecture Notes in Business Information Processing (LNBIP).

1 [smm-maker-profile](https://www.jsclasses.org/smm-maker-profile) (accessed August 25, 2019).
ABISHEV, A.; GARIMELLA, V. R. K.; GARCIA, D.; WEBER, I. Who watches (and shares) what on youtube? and when?: Using twitter to understand youtube viewership. In: Proceedings of the 7th ACM International Conference on Web Search and Data Mining. New York, NY, USA: ACM, 2014. (WSDM ’14), p. 593–602. ISBN 978-1-4503-2351-2. Citation on page 59.

AGGARWAL, C. C.; WOLF, J. L.; YU, P. S.; PROCOPIUC, C.; PARK, J. S. Fast algorithms for projected clustering. In: Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data. New York, NY, USA: ACM, 1999. (SIGMOD ’99), p. 61–72. ISBN 1-58113-084-8. Available: <http://doi.acm.org/10.1145/304182.304188>. Citation on page 34.

AGRAWAL, R.; GEHRKE, J.; GUNOPULOS, D.; RAGHAVAN, P. Automatic subspace clustering of high dimensional data for data mining applications. SIGMOD Rec., ACM, New York, NY, USA, v. 27, n. 2, p. 94–105, Jun. 1998. ISSN 0163-5808. Available: <http://doi.acm.org/10.1145/276305.276314>. Citation on page 34.

AGRAWAL, R.; IMIEL´nSKI, T.; SWAMI, A. Mining association rules between sets of items in large databases. SIGMOD Rec., ACM, New York, NY, USA, v. 22, n. 2, p. 207–216, Jun. 1993. ISSN 0163-5808. Citation on page 34.

AGRAWAL, R.; SRIKANT, R. et al. Fast algorithms for mining association rules. In: Proc. 20th int. conf. very large data bases, VLDB. [S.I.: s.n.], 1994. v. 1215, p. 487–499. Citations on pages 34 and 91.

AL-GARADI, M. A.; VARATHAN, K. D.; RAVANA, S. D.; AHMED, E.; MUJTABA, G.; KHAN, M. U. S.; KHAN, S. U. Analysis of online social network connections for identification of influential users: Survey and open research issues. ACM Comput. Surv., ACM, New York, NY, USA, v. 51, n. 1, p. 16:1–16:37, Jan. 2018. ISSN 0360-0300. Available: <http://doi.acm.org/10.1145/3155897>. Citations on pages 59 and 60.

AREEKIJSEREE, K.; LAISHRAM, R.; SOUNDARAJAN, S. Guidelines for online network crawling: A study of data collection approaches and network properties. In: Proceedings of the 10th ACM Conference on Web Science. New York, NY, USA: ACM, 2018. (WebSci ’18), p. 57–66. ISBN 978-1-4503-5563-6. Available: <http://doi.acm.org/10.1145/3201064.3201066>. Citations on pages 49 and 50.

AUNG, M.; BONOMETTI, V.; DRACHEN, A.; COWLING, P.; KOKKINAKIS, A. V.; YODER, C.; WADE, A. Predicting skill learning in a large, longitudinal moba dataset. In: 2018 IEEE Conference on Computational Intelligence and Games (CIG). [S.I.: s.n.], 2018. p. 1–7. ISSN 2325-4289. Citations on pages 44 and 45.

BARABÁSI, A.-L.; PÓSFAI, M. Network science. Cambridge, USA: Cambridge university press, 2016. ISBN 9781107076266. Citations on pages 36, 39, 40, 45, and 59.

BEAUPRE, S.; WILES, T.; BRIGGS, S.; SMITH, G. A design pattern approach for multi-game level generation. In: Fourteenth Artificial Intelligence and Interactive Digital Entertainment Conference. [S.I.: s.n.], 2018. Citations on pages 76, 77, 80, and 94.
BENNIS, L.; BENHLIMA, S. Comparative study of the process model of serious game design through the generic model dice. In: 2015 Intelligent Systems and Computer Vision (ISCV). [S.l.: s.n.], 2015. p. 1–5. Citation on page 77.

BRIN, S.; MOTWANI, R.; ULLMAN, J. D.; TSUR, S. Dynamic itemset counting and implication rules for market basket data. In: Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data. New York, NY, USA: ACM, 1997. (SIGMOD ’97), p. 255–264. ISBN 0-89791-911-4. Citation on page 35.

BROWNE, C.; LIAPIS, A.; WINANDS, M. Special issue on computer aided game and puzzle design. ICGA Journal, IOS Press, n. Preprint, p. 1–2, 2019. Citations on pages 76 and 77.

CALINSKI, T.; HARABASZ, J. A dendrite method for cluster analysis. Communications in Statistics, Taylor & Francis, v. 3, n. 1, p. 1–27, 1974. Citation on page 34.

CHINO, D. Y. T.; COSTA, A. F.; TRAINA, A. J. M.; FALOUTSOS, C. Volttime: Unsupervised anomaly detection on users’ online activity volume. In: SIAM. Proceedings of the 2017 SIAM International Conference on Data Mining. SIAM International Conference on Data Mining, 2017. p. 108–116. Available: <https://epubs.siam.org/doi/abs/10.1137/1.9781611974973.13>. Citation on page 60.

CLASSE, T. M. d.; ARAUJO, R. M. D.; XEXéO, G. B.; SIQUEIRA, S. The play your process method for business process-based digital game design. International Journal of Serious Games, v. 6, n. 1, p. 27–48, Mar. 2019. Citation on page 77.

CLAUSET, A.; SHALIZI, C.; NEWMAN, M. Power-law distributions in empirical data. SIAM Review, v. 51, n. 4, p. 661–703, 2009. Citation on page 54.

COPPIN, B. Artificial Intelligence Illuminated. USA: Jones and Bartlett Publishers, Inc., 2004. ISBN 0763732303. Citation on page 31.

CORDEIRO, R. L. F.; TRAINA, A. J. M.; FALOUTSOS, C.; JR., C. T. Halite: Fast and scalable multiresolution local-correlation clustering. IEEE Trans. Knowl. Data Eng., v. 25, n. 2, p. 387–401, 2013. Available: <https://doi.org/10.1109/TKDE.2011.176>. Citation on page 34.

COVER, T.; HART, P. Nearest neighbor pattern classification. IEEE Transactions on Information Theory, v. 13, n. 1, p. 21–27, January 1967. ISSN 0018-9448. Citation on page 33.

DAVIES, D. L.; BOULDIN, D. W. A cluster separation measure. IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-1, n. 2, p. 224–227, April 1979. ISSN 0162-8828. Citation on page 34.

DUNN, J. C. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. Journal of Cybernetics, Taylor & Francis, v. 3, n. 3, p. 32–57, 1973. Citation on page 34.
EBERHARD, L.; KASPER, P.; KONCAR, P.; GüTL, C. Investigating helpfulness of video game reviews on the steam platform. In: 2018 Fifth International Conference on Social Networks Analysis, Management and Security (SNAMS). [S.l.: s.n.], 2018. p. 43–50. Citations on pages 28, 43, and 45.

ESTER, M.; KRIEGEL, H.-P.; SANDER, J.; XU, X. A density-based algorithm for discovering clusters in large spatial databases with noise. In: . [S.l.]: AAAI Press, 1996. p. 226–231. Citation on page 34.

FAYYAD, U. M.; PIATETSKY-SHAapiro, G.; SMYTH, P.; UTHURUSAMY, R. Advances in knowledge discovery and data mining. [S.l.]: AAAI press Menlo Park, 1996. Citation on page 32.

FUNGE, J. D. Artificial intelligence for computer games: an introduction. [S.l.]: Peters Corp., 2004. Hardcover. ISBN 1568812086. Citation on page 31.

GOLDSCHMIDT, R.; PASSOS, E.; BEZERRA, E. Data Mining. Brasil: Elsevier Brasil, 2015. Citations on pages 32, 33, 36, and 40.

GROS, D.; HACKENHOLT, A.; ZAWADZKI, P.; WANNER, B. Interactions of twitch users and their usage behavior. In: MEISELWITZ, G. (Ed.). Social Computing and Social Media. Technologies and Analytics. Cham: Springer International Publishing, 2018. p. 201–213. ISBN 978-3-319-91485-5. Citation on page 57.

GUZDIAL, M.; LIAO, N.; CHEN, J.; CHEN, S.-Y.; SHAH, S.; SHAH, V.; RENO, J.; SMITH, G.; RIEDL, M. O. Friend, collaborator, student, manager: How design of an ai-driven game level editor affects creators. In: Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems. New York, NY, USA: ACM, 2019. (CHI ’19), p. 624:1–624:13. ISBN 978-1-4503-5970-2. Citations on pages 75, 76, and 77.

GUZDIAL, M.; RIEDL, M. Automated game design via conceptual expansion. In: AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment. [s.n.], 2018. Available: <https://www.aaai.org/ocs/index.php/AIIDE/AIIDE18/paper/view/18108>. Citations on pages 75, 76, and 77.

HILVERT-BRUCE, Z.; NEILL, J. T.; SJöBLOM, M.; HAMARI, J. Social motivations of live-streaming viewer engagement on twitch. Computers in Human Behavior, v. 84, p. 58 – 67, 2018. ISSN 0747-5632. Available: <http://www.sciencedirect.com/science/article/pii/S0747563218300712>. Citation on page 57.

HUIZINGA, J. Homo ludens: A study of the Play-Element in Culture. [S.l.]: Boston, MA: The Beacon Press, 1955. Citation on page 27.

JIA, A. L.; SHEN, S.; BOVENKAMP, R. V. D.; IOSUP, A.; KUIPERS, F.; EPEMA, D. H. J. Socializing by gaming: Revealing social relationships in multiplayer online games. ACM Trans. Knowl. Discov. Data, ACM, New York, NY, USA, v. 10, n. 2, p. 11:1–11:29, Oct. 2015. ISSN 1556-4681. Available: <http://doi.acm.org/10.1145/2736698>. Citations on pages 28, 43, and 45.

JOHNSON, M.; HOFMANN, K.; HUTTON, T.; BIGNELL, D. The malmo platform for artificial intelligence experimentation. In: IJCAI. AAAI Press, Palo Alto, California USA: [s.n.], 2016. p. 4246–4247. Available: <https://github.com/Microsoft/malmo>. Citation on page 44.
KAMBER, M.; HAN, J.; PEI, J. *Data mining: Concepts and techniques*. Amsterdam: Elsevier, 2012. Citations on pages 28, 32, 33, 35, and 63.

KARAVOLOS, D.; LIAPIS, A.; YANNAKAKIS, G. N. Using a surrogate model of gameplay for automated level design. In: 2018 IEEE Conference on Computational Intelligence and Games (CIG). [S.l.: s.n.], 2018. p. 1–8. ISSN 2325-4289. Citation on page 77.

KARPOUZIS, K.; YANNAKAKIS, G. N.; SHAKER, N.; ASTERIADIS, S. The platformer experience dataset. In: 2015 International Conference on Affective Computing and Intelligent Interaction. USA: ACII, 2015. p. 712–718. Citations on pages 44 and 45.

KOZA, J. R. *Genetic programming: On the programming of computers by means of natural selection*. MA. [S.l.]: MIT Press, Cambridge, 1992. Citation on page 33.

KUNZE., L. F.; AMARAL., T.; MORAES., L. M. P.; OLIVEIRA., J. J. M.; JUNIOR., A. G. B.; SOUSA., E. P. M. de; CORDEIRO., R. L. F. Classification analysis of ndvi time series in metric spaces for sugarcane identification. In: INSTICC. *Proceedings of the 20th International Conference on Enterprise Information Systems - Volume 1: ICEIS*, [S.l.]: SciTePress, 2018. p. 162–169. ISBN 978-989-758-298-1. Citation on page 99.

LEE, J.; LEE, M.; CHOI, I. H. Social network games uncovered: Motivations and their attitudinal and behavioral outcomes. *Cyberpsychology, Behavior, and Social Networking*, v. 15, n. 12, p. 643–648, 2012. PMID: 23020746. Available: <https://doi.org/10.1089/cyber.2012.0093>. Citation on page 28.

LEE, Y.-T.; CHEN, K.-T.; CHENG, Y.-M.; LEI, C.-L. World of warcraft avatar history dataset. In: *Proceedings of the Second Annual ACM Conference on Multimedia Systems*. New York, NY, USA: ACM, 2011. (MMSys ’11), p. 123–128. ISBN 978-1-4503-0518-1. Available: <http://doi.acm.org/10.1145/1943552.1943569>. Citations on pages 43, 44, and 45.

LIM, C.-U.; HARRELL, D. F. Comparing clustering approaches for modeling players’ values through avatar construction. *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2015. Available: <https://www.aaai.org/ocs/index.php/AIIDE/AIIDE15/paper/view/11591>. Citation on page 44.

LIN, Z.; GEHRING, J.; KHALIDOV, V.; SYNNAEVE, G. Stardata: A starcraft ai research dataset. *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2017. Available: <https://www.aaai.org/ocs/index.php/AIIDE/AIIDE17/paper/view/15837>. Citations on pages 44 and 45.

LIU, N.; LI, L.; XU, G.; YANG, Z. Identifying domain-dependent influential microblog users: A post-feature based approach. In: *AAAI*. [S.l.: s.n.], 2014. p. 3122–3123. Citation on page 60.

LLOYD, S. Least squares quantization in pcm. *IEEE transactions on information theory*, IEEE, v. 28, n. 2, p. 129–137, 1982. Citation on page 34.

LUCAS, S. M. Computational intelligence and ai in games: A new ieee transactions. *IEEE Transactions on Computational Intelligence and AI in Games*, v. 1, n. 1, p. 1–3, March 2009. ISSN 1943-068X. Citations on pages 27, 31, and 55.

MANDELBROT, B. The pareto-levy law and the distribution of income. *International Economic Review*, JSTOR, v. 1, n. 2, p. 79–106, 1960. Citation on page 85.
METOYER, R.; STUMPF, S.; NEUMANN, C.; DODGE, J.; CAO, J.; SCHNABEL, A. Explaining how to play real-time strategy games. *Knowledge-Based Systems*, v. 23, n. 4, p. 295 – 301, 2010. ISSN 0950-7051. Artificial Intelligence 2009. Available: <http://www.sciencedirect.com/science/article/pii/S0950705109001488>. Citation on page 55.

MITZENMACHER, M. A brief history of generative models for power law and lognormal distributions. *Internet Mathematics*, Taylor & Francis, v. 1, n. 2, p. 226–251, 2004. Citation on page 54.

MOISE, G.; SANDER, J. Finding non-redundant, statistically significant regions in high dimensional data: A novel approach to projected and subspace clustering. In: *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM, 2008. (KDD ’08), p. 533–541. ISBN 978-1-60558-193-4. Available: <http://doi.acm.org/10.1145/1401890.1401956>. Citation on page 34.

MOISE, G.; SANDER, J.; ESTER, M. P3c: A robust projected clustering algorithm. In: *Sixth International Conference on Data Mining (ICDM’06)*. [S.l.: s.n.], 2006. p. 414–425. ISSN 1550-4786. Citations on pages 34 and 88.

MORAES, L. M. P.; CORDEIRO, R. L. F. Detecting influencers in very large social networks of games. In: INSTICC. *Proceedings of the 21st International Conference on Enterprise Information Systems - Volume 1: ICEIS*, Crete, Greece: SciTePress, 2019. p. 93–103. ISBN 978-989-758-372-8. Citations on pages 98 and 99.

_____. Smmnet: A social network of games dataset. In: BRAZILIAN COMPUTER SOCIETY (SBC). *Brazilian Symposium on Databases (SBBD) 2019 - Dataset Showcase Workshop (DSW)*. Ceará, Brazil, 2019. Citation on page 98.

MORONE, F.; MIN, B.; BO, L.; MARI, R.; MAKSE, H. A. Collective influence algorithm to find influencers via optimal percolation in massively large social media. *Scientific reports*, Nature Publishing Group, v. 6, p. 30062, 2016. Citations on pages 59 and 60.

NEWMAN, M. Power laws, pareto distributions and zipf’s law. *Contemporary Physics*, Taylor & Francis, v. 46, n. 5, p. 323–351, 2005. Citations on pages 53 and 85.

PAGE, L.; BRIN, S.; MOTWANI, R.; WINograd, T. *The PageRank Citation Ranking: Bringing Order to the Web*. [S.l.], 1999. Previous number = SIDL-WP-1999-0120. Available: <http://ilpubs.stanford.edu:8090/422/>. Citation on page 40.

PARK, J. S.; CHEN, M.-S.; YU, P. S. An effective hash-based algorithm for mining association rules. *SIGMOD Rec.*, ACM, New York, NY, USA, v. 24, n. 2, p. 175–186, May 1995. ISSN 0163-5808. Available: <http://doi.acm.org/10.1145/568271.223813>. Citation on page 34.

PEARL, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. [S.l.]: Morgan Kaufmann Publishers Inc., 1988. Citation on page 33.

PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETENNOFER, P.; WEISS, R.; DUBOURG, V.; VANDERPLAS, J.; PASSOS, A.; COURNAPEAU, D.; BRUCHER, M.; PERROT, M.; DUCHESNAY, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011. Citation on page 69.
PEI, S.; MORONE, F.; MAKSE, H. A. Theories for influencer identification in complex networks. In: ____. Complex Spreading Phenomena in Social Systems: Influence and Contagion in Real-World Social Networks. Cham: Springer International Publishing, 2018. p. 125–148. ISBN 978-3-319-77332-2. Available: <https://doi.org/10.1007/978-3-319-77332-2_8>. Citation on page 57.

PIATETSKY-SHAPIRO, G. Discovery, analysis, and presentation of strong rules. Knowledge discovery in databases, Menlo Park, CA: AAI/MIT, p. 229–238, 1991. Citation on page 35.

PIVEC, M.; KEARNEY, P. Games for learning and learning from games. Organizacija, v. 40, n. 6, 2007. ISSN 1581-1832. Available: <http://organizacija.fov.uni-mb.si/index.php/organizacija/article/view/208>. Citation on page 27.

QI, L.; HUANG, Y.; LI, L.; XU, G. Learning to rank domain experts in microblogging by combining text and non-text features. In: 2015 International Conference on Behavioral, Economic and Socio-cultural Computing (BESC). [S.I.: s.n.], 2015. p. 28–31. Citation on page 60.

QUINLAN, J. R. C4.5: Programming for machine learning. Morgan Kauffmann, v. 38, p. 48, 1993. Citation on page 33.

ROUSSEEUW, P. J. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics, v. 20, p. 53–65, 1987. ISSN 0377-0427. Citation on page 34.

SALEN, K.; TEKINBAŞ, K. S.; ZIMMERMAN, E. Rules of play: Game design fundamentals. [S.I.]: MIT press, 2004. Citation on page 75.

SAVIĆ, M.; IVANOVIĆ, M.; JAIN, L. C. Introduction to complex networks. In: ____. Complex Networks in Software, Knowledge, and Social Systems. Cham: Springer International Publishing, 2019. p. 3–16. ISBN 978-3-319-91196-0. Available: <https://doi.org/10.1007/978-3-319-91196-0_1>. Citation on page 45.

SJöBLOM, M.; HAMARI, J. Why do people watch others play video games? an empirical study on the motivations of twitch users. Computers in Human Behavior, v. 75, p. 985–996, 2017. ISSN 0747-5632. Available: <http://www.sciencedirect.com/science/article/pii/S0747563216307208>. Citation on page 57.

STRANG, G.; STRANG, G.; STRANG, G.; STRANG, G. Introduction to Linear Algebra. [S.I.]: Wellesley-Cambridge Press Wellesley, MA, 2016. ISBN 978-09802327-7-6. Citation on page 62.

WANG, X.; ZHANG, X.; YI, D.; ZHAO, C. Identifying influential spreaders in complex networks through local effective spreading paths. Journal of Statistical Mechanics: Theory and Experiment, v. 2017, n. 5, p. 053402, 2017. Available: <http://stacks.iop.org/1742-5468/2017/i=5/a=053402>. Citations on pages 59 and 60.

WERBOS, P. Beyond regression: New tools for prediction and analysis in the behavioral sciences. Ph. D. dissertation, Harvard University, 1974. Available: <https://ci.nii.ac.jp/naid/10004070196/en/>. Citation on page 33.

WESTABY, J. D. Dynamic network theory: How social networks influence goal pursuit. [S.I.]: American Psychological Association, 2012. Citations on pages 45 and 58.
XU, R.; WUNSCH, D. *Clustering*. USA: IEEE Press Series on Computational Intelligence - John Wiley & Sons, 2009. Citation on page 34.

YANNAKAKIS, G. N. Game ai revisited. In: *Proceedings of the 9th Conference on Computing Frontiers*. New York, NY, USA: ACM, 2012. (CF ’12), p. 285–292. ISBN 978-1-4503-1215-8. Available: <http://doi.acm.org/10.1145/2212908.2212954>. Citations on pages 27, 28, 35, 44, and 55.

YANNAKAKIS, G. N.; SPRONCK, P.; LOIACONO, D.; ANDRÉ, E. Player modeling. In: LUCAS, S. M.; MATEAS, M.; PREUSS, M.; SPRONCK, P.; TOGELIUS, J. (Ed.). *Artificial and Computational Intelligence in Games*. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2013, (Dagstuhl Follow-Ups, v. 6), p. 45–59. ISBN 978-3-939897-62-0. Citations on pages 28, 31, 35, 36, 43, and 44.

YANNAKAKIS, G. N.; TOGELIUS, J. A panorama of artificial and computational intelligence in games. *IEEE Transactions on Computational Intelligence and AI in Games*, v. 7, n. 4, p. 317–335, Dec 2015. ISSN 1943-068X. Citations on pages 28, 31, 36, 43, 44, and 75.

—. *Artificial Intelligence and Games*. Cham: Springer International Publishing, 2018. 3–28 p. ISBN 978-3-319-63519-4. Available: <https://doi.org/10.1007/978-3-319-63519-4_1>. Citations on pages 27, 28, 31, 32, 35, 36, 43, 75, 76, and 77.

ZOELLER, G. Game development telemetry in production. In: —. *Game Analytics: Maximizing the Value of Player Data*. London: Springer London, 2013. p. 111–135. ISBN 978-1-4471-4769-5. Available: <https://doi.org/10.1007/978-1-4471-4769-5_7>. Citation on page 44.
This appendix details our proposed feature aggregation procedure. It follows the original discussion from Section 5.2.4.

\[ F_{<I,\{\text{powerUp}\}} = F_{\text{Mushroom}} + F_{\text{UpMushroom}} + F_{\text{FireFlower}} + F_{\text{SuperStar}} + F_{\text{SuperItem}} + F_{\text{WeirdMushroom}} + F_{\text{MegaMushroom}} \]

\[ F_{<I,\{\text{vehicle}\}} = F_{\text{LakituCloud}} + F_{\text{KoopaClown}} + F_{\text{FireKoopaClown}} + F_{\text{YoshiEgg}} \]

\[ F_{<I,\{\text{support}\}} = F_{\text{Block}} + F_{\text{AirSign}} + F_{\text{CheckpointFlag}} \]

\[ F_{<I,\{\text{jump}\}} = F_{\text{Bumper}} + F_{\text{JumpStep}} + F_{\text{NoteBlock}} \]

\[ F_{<I,\{\text{move}\}} = F_{\text{Lift}} + F_{\text{FallLift}} + F_{\text{LavaLift}} + F_{\text{BlueLavaLift}} + F_{\text{BeltSprint}} + F_{\text{VineBody}} + F_{\text{OneWayGate}} \]

\[ F_{<I,\{\text{teleport}\}} = F_{\text{Door}} + F_{\text{PDoor}} + F_{\text{KeyDoor}} + F_{\text{Pipe}} \]

\[ F_{<I,\{\text{puzzle}\}} = F_{\text{PSwitch}} + F_{\text{Key}} + F_{\text{PowBlock}} \]

\[ F_{<I,\{\text{bonus}\}} = F_{\text{Coin}} + F_{\text{RedCoin}} \]

\[ F_{<C,\{\text{fragile,walk}\}} = F_{\text{Goomba}} + F_{\text{GreenKoopaTroopa}} + F_{\text{RedKoopaTroopa}} + F_{\text{BuzzyBeetle}} + F_{\text{DryBones}} + F_{\text{CheepCheep}} + F_{\text{DeepCheep}} + F_{\text{MontyMole}} \]

\[ F_{<C,\{\text{fragile,walk,\big}\}} = F_{\text{GoombaBig}} + F_{\text{GreenKoopaTroopaBig}} + F_{\text{RedKoopaTroopaBig}} + F_{\text{BuzzyBeetleBig}} + F_{\text{DryBonesBig}} + F_{\text{CheepCheepBig}} + F_{\text{DeepCheepBig}} + F_{\text{MontyMoleBig}} \]

\[ F_{<C,\{\text{fragile,\fly}\}} = F_{\text{GoombaWing}} + F_{\text{GreenKoopaTroopaWing}} + F_{\text{RedKoopaTroopaWing}} + F_{\text{BuzzyBeetleWing}} + F_{\text{DryBonesWing}} + F_{\text{Bloomer}} + F_{\text{BloomerWing}} + F_{\text{BloomerNanny}} + F_{\text{BloomerNannyWing}} + F_{\text{MontyMoleWing}} + F_{\text{CheepCheepWing}} + F_{\text{DeepCheepWing}} \]
APPENDIX A. Feature Aggregation

\[ F_{<C,\{\text{fragile, fly, big}\}} = F_{\text{GoombaBigWing}} + F_{\text{GreenKoopaTroopaBigWing}} + F_{\text{RedKoopaTroopaBigWing}} + \]
\[ F_{\text{BuzzyBeetleBigWing}} + F_{\text{DryBonesBigWing}} + F_{\text{BlooperBig}} + F_{\text{BlooperBigWing}} + F_{\text{BlooperNannyBig}} + \]
\[ F_{\text{BlooperNannyBigWing}} + F_{\text{CheepCheepBigWing}} + F_{\text{DeepCheepBigWing}} + F_{\text{MontyMoleBigWing}} \]

\[ F_{<C,\{\text{fragile, walk, shooter}\}} = F_{\text{HammerBro}} + F_{\text{Bombhei}} + F_{\text{ActiveBombhei}} + F_{\text{Kameck}} + F_{\text{RockyWrench}} \]

\[ F_{<C,\{\text{fragile, walk, shooter, big}\}} = F_{\text{HammerBroBig}} + F_{\text{BombheiBig}} + F_{\text{ActiveBombheiBig}} + F_{\text{KameckBig}} + F_{\text{RockyWrenchBig}} \]

\[ F_{<C,\{\text{fragile, fly, shooter}\}} = F_{\text{HammerBroWing}} + F_{\text{BombheiWing}} + F_{\text{ActiveBombheiWing}} + F_{\text{KameckWing}} + F_{\text{RockyWrenchWing}} + F_{\text{Lakitu}} + F_{\text{LakituWing}} \]

\[ F_{<C,\{\text{fragile, fly, shooter, big}\}} = F_{\text{HammerBroBigWing}} + F_{\text{BombheiBigWing}} + F_{\text{ActiveBombheiBigWing}} + F_{\text{KameckBigWing}} + F_{\text{RockyWrenchBigWing}} + F_{\text{LakituBig}} + F_{\text{LakituBigWing}} \]

\[ F_{<C,\{\text{robust, stable}\}} = F_{\text{PiranhaPlant}} + F_{\text{Skewer}} + F_{\text{SkewerWing}} + F_{\text{Tsuta}} + F_{\text{CirclingBoo}} + F_{\text{CirclingBooWing}} + F_{\text{Burner}} + F_{\text{Muncher}} + F_{\text{ChainChomp}} + F_{\text{ChainChompWing}} + F_{\text{Thwomp}} + F_{\text{ThwompWing}} + F_{\text{FireBar}} + F_{\text{SpikeTopBlock}} \]

\[ F_{<C,\{\text{robust, stable, big}\}} = F_{\text{PiranhaPlantBig}} + F_{\text{SkewerBig}} + F_{\text{SkewerBigWing}} + F_{\text{CirclingBooBig}} + F_{\text{CirclingBooBigWing}} + F_{\text{MuncherBig}} + F_{\text{ChainChompBig}} + F_{\text{ChainChompBigWing}} + F_{\text{Grinder}} + F_{\text{ThwompBig}} + F_{\text{ThwompBigWing}} \]

\[ F_{<C,\{\text{robust, walk}\}} = F_{\text{Spiny}} + F_{\text{BlueSpiny}} + F_{\text{SpikeTop}} + F_{\text{BlueSpikeTop}} + F_{\text{Wiggler}} + F_{\text{AngryWiggler}} \]

\[ F_{<C,\{\text{robust, walk, big}\}} = F_{\text{SpinyBig}} + F_{\text{BlueSpinyBig}} + F_{\text{SpikeTopBig}} + F_{\text{BlueSpikeTopBig}} + F_{\text{WigglerBig}} + F_{\text{AngryWigglerBig}} \]

\[ F_{<C,\{\text{robust, fly}\}} = F_{\text{PiranhaPlantWing}} + F_{\text{SpinyWing}} + F_{\text{BlueSpinyWing}} + F_{\text{SpikeTopWing}} + F_{\text{BlueSpikeTopWing}} + F_{\text{Boo}} + F_{\text{BooWing}} + F_{\text{WigglerWing}} + F_{\text{AngryWigglerWing}} + F_{\text{LavaBubble}} + F_{\text{LavaBubbleWing}} + F_{\text{MuncherWing}} \]

\[ F_{<C,\{\text{robust, fly, big}\}} = F_{\text{PiranhaPlantBigWing}} + F_{\text{SpinyBigWing}} + F_{\text{BlueSpinyBigWing}} + F_{\text{SpikeTopBigWing}} + F_{\text{BlueSpikeTopBigWing}} + F_{\text{BooBig}} + F_{\text{BooBigWing}} + F_{\text{WigglerBigWing}} + F_{\text{AngryWigglerBigWing}} + F_{\text{LavaBubbleBig}} + F_{\text{LavaBubbleBigWing}} + F_{\text{MuncherBigWing}} \]

\[ F_{<C,\{\text{robust, stable, shooter}\}} = F_{\text{FirePiranhaPlant}} + F_{\text{Cannon}} + F_{\text{RedCannon}} + F_{\text{BillBlaster}} + F_{\text{BullEyeBlaster}} + F_{\text{Bowser}} + F_{\text{BowserJr}} \]

\[ F_{<C,\{\text{robust, stable, shooter, big}\}} = F_{\text{FirePiranhaPlantBig}} + F_{\text{BowserBig}} + F_{\text{BowserJrBig}} \]

\[ F_{<C,\{\text{robust, fly, shooter}\}} = F_{\text{FirePiranhaPlantWing}} + F_{\text{BowserWing}} + F_{\text{BowserJrWing}} \]

\[ F_{<C,\{\text{robust, fly, shooter, big}\}} = F_{\text{FirePiranhaPlantBigWing}} + F_{\text{BowserBigWing}} + F_{\text{BowserJrBigWing}} \]
