# Rational Agent Architecture to Recommend which Item to Buy in MOBA Videogames


**Juan Guillermo López Guzmán**
**DIRECTOR: Cesar Julio Bustacara Medina**

**PONTIFICA UNIVERSIDAD JAVERIANA**
**FACULTAD DE INGENIERIA**
**MAESTRIA EN INTELIGENCIA ARTIFICIAL**
**BOGOTA DC, COLOMBIA**
**NOVIEMBRE, 2021**

# Rational Agent Architecture to Recommend which Item to Buy in League of Legends

Juan Guillermo López Guzmán, César Julio Bustacara Medina

***Resumen*— Los videojuegos multijugador de arena de batalla en línea (MOBA), es un genero de videojuegos que durante la última década han ganado popularidad en la escena competitiva de los E-Sports. Este incremento en su popularidad y la complejidad propia de los mismos han llamado la atención de investigadores en todas las áreas del conocimiento, incluyendo la Inteligencia Artificial. Dichos investigadores han utilizado una amplia variedad de técnicas de Aprendizaje de Maquina buscando mejorar la experiencia de diversos usuarios -jugadores novatos, jugadores expertos, espectadores, entre otros- a través de modelos de predicción, sistemas de recomendación y, aunque se han utilizado técnicas de optimización; estas últimas han sido las menos utilizadas en los videojuegos tipo MOBA. Por ello, el presente trabajo de investigación propone la arquitectura de un agente racional capaz de recomendar a un jugador que objeto comprar para aumentar sus probabilidades de ganar una partida, utilizando una técnica de optimización para la generación de recomendaciones. En la arquitectura propuesta, el agente percibe su ambiente con la información disponible en el API del videojuego League of Legends -uno de los MOBA mas populares actualmente-. Tal información es interpretada por una Regresión Logística que durante las etapas tempranas del juego demostró tener una precisión alrededor de 0.975. A su vez, la técnica de optimización seleccionada para generar la sugerencia fue GRASP; en promedio cada sugerencia es generada en 0.36 segundos, estas sugerencias durante la experimentación lograron aumentar la probabilidad de ganar una partida en promedio 5.2x.**

***Abstract*— Multiplayer online battle arena (MOBA) video games are a genre of video games that during the last decade have gained popularity in the competitive E-Sports scene. This increase in popularity and MOBA's complexity have attracted the attention of researchers in all areas of knowledge, including Artificial Intelligence (AI). AI researchers have used a wide variety of Machine Learning techniques seeking to improve the experience of various users - novice players, expert players, spectators, among others - through prediction models, recommendation systems and optimization algorithms. However, optimization algorithms have been the least used in MOBA videogames. For that reason, this research proposes the architecture of a rational agent capable of recommending to a player what item to buy to increase his probabilities of winning a game, using an optimization technique for generating recommendations. In the proposed architecture, the agent perceives his environment with the information available in the API of League of Legends -currently, one of the most popular MOBA videogames -. Such information is interpreted by a Logistic Regression that during the early stages of the game was shown to have an accuracy around 0.975. Additionally, the optimization technique selected to generate the suggestion was GRASP. On average each suggestion is generated in 0.36 seconds. During experimentation, these suggestions increase the probability of winning a game on average 5.2x.**

***Impact Statement* — In 2021 League of Legends has reached a total of 115 million active users. All these players face decisions with a high complexity during a match. For instance, players could buy until six items per match from a pool higher than three hundred options. It means, players buying items have a spectrum of decisions higher than $6.9 * 10^{14}$. Tools that help players to take this decision reducing the uncertainty, would impact positively the user experience. Novice players could accelerate their learning curve and expert players could discover new item configurations. It would increase the overall level of the game and, in consequence, the level in competitive matches.**

***Index Terms*— Autonomous systems, Rational agent, Supervised learning, Machine learning, Data Mining, Predictive models, Pattern recognition, Logistic Regression, Optimization, Greedy algorithms.**

## I. Introduction

VIDEOGAMES are already present in our everyday life. They have reached a massive audience from all ages due to their wide variety of genders and thematic. Those genders evolve trough time and originate new ones, for instance, Real Time Strategy games (RTS) originates at the early 2000's a different sub-gender known as Massive Online Battle Arena (MOBA) [1].

There are many games that could be considered as MOBA predecessors. Although, the first videogame formally classified as a MOBA was Defense of the Ancients (DotA). It was created in 2003 by players of the RPG videogame Warcraft III. DotA is based on the Warcraft III lore, however, it has specific gameplay characteristics.

Each match starts with two adversary teams, typically conformed of five players. [2]. Usually, both teams have main structures that are located at the opposite corners of the

This paragraph of the first footnote will contain the date on which you submitted your paper for review. It will also contain support information, including sponsor and financial support acknowledgment. For example, "This work was supported in part by the U.S. Department of Commerce under Grant BS123456."

J.G. López-Guzmán, student of the MSc in Artificial Intelligence at the Pontificia Universidad Javeriana, Bogotá, Colombia. (e-mail: juang.lopezg@javeriana.edu.co).

C.J. Bustacara-Medina, Computer Science Department, Pontifica Universidad Javeriana, Bogotá, Colombia. (e-mail: cbustaca@javeriana.edu.co).

This paragraph will include the Associate Editor who handled your paper.

battlefield. The first team to destroy the adversaries' main structure wins the match [2]. Destroying other structures within the adversary team's base may provide other benefits. Defensive structures, which are usually automatic 'towers', are in place to prevent the base destruction. Each team is assisted by relatively weak computer-controlled units, called 'minions', that periodically spawn in groups at both bases, marching down predefined paths (called 'lanes') toward their enemy base. Players can aid them, turning the minions into a useful army for striking the adversaries' defenses [3]. There are typically three (3) 'lanes' on the battlefield that are the main ways of getting from one base to another. The lanes are known as top, middle and bottom lane, or, in gamer shorthand – 'top', 'mid' and 'bot'. Between the lanes is an uncharted area called 'jungle' [4]. The 'jungle' is a territory to neutral monsters that are hostile to both teams and appear in marked locations on the map known as 'camps' [5]. Defeating neutral monsters brings various benefits to the players and their team, such as growth in power, buffs, or assistance in pushing the lane [6].

In 2009 and 2012, respectively, the most popular MOBA videogames were born: League of Legends (LoL) and Defense of the Ancients II (DotA II). They maintain the gameplay described above: Two teams of five players must destroy the nexus of the enemy team located in the different side of a predefined map. Fighting against defensive structures, player, and non-player characters.

Both achieved great popularity in the e-sports scenario. For instance, 2019 League of Legends World Championship got an online audience greater than one hundred million through several electronic platforms as Twitch [7]. Also, it got an in-site audience near to forty thousand people on the Shanghai Olympic stadium.

Their popularity has attracted a lot of researchers' attention from various knowledge fields as psychology [8], marketing [9], computer science and artificial intelligence. Specially, Artificial Intelligence researchers have found in MOBA videogames an interesting playground for the application of a wide spectrum of machine learning techniques. They have used techniques to predict an outcome such as whether a team will win a match [10], to generate a recommendation of an in-game action [11], and to find the best solution from all feasible options [12]. These techniques have also been used as components in rational agents which help players [13] or spectators [14] to improve their experience.

An extensive number of papers applying machine learning techniques were reviewed and analyzed. This literature review allows to identify relevant insights such as: Most common use cases for machine learning algorithm, most popular machine learning techniques used and their results, independent variables frequently used in MOBA videogames applications, and how this information have been used into rational agents. Most important, this exhaustive research was helpful to identify unsolved questions on the application of rational agents in MOBA videogames.

Based on these questions the goal of this paper is: To propose the architecture for a rational agent which uses machine learning techniques to perceive the current in-game situation and suggest what items to buy during a game in order to maximize the chance to win a match, taking in consideration the available in-game gold of the player as a constraint. In consequence, the item recommendation will be gotten using an optimization algorithm.

For that purpose: Section II describes previous works research, Section III details the methodology required to achieve the investigation objective, Section IV introduces the conducted experiments, Section V describes conclusions, and finally Section VI presents potential future works related to this paper.

## II. Background

MOBA videogames have attracted the attention of researchers from several disciplines during the last years. One of these disciplines is Artificial Intelligence. AI researchers have explored how to introduce rational agents which improve game experience or performance of players. Because during a match players should consider a lot of variables to take real-time decisions from a wide number of possibilities. It makes possible to experiment with a variety of techniques looking the way to reduce the uncertainty of in-game decisions without taking a lot of processing time.

An extensive review of papers applying artificial intelligence and machine learning techniques into the MOBA videogames context was performed. It allowed to summarize and understand what techniques have been used, and how they have been implemented into rational agents. Also, this literature review was helpful to identify unsolved questions on the application of rational agents in MOBA videogames.

Consequently, Section A will describe machine learning techniques used in MOBA videogames context. Section B will introduce research that have used machine learning techniques to apply rational agents capable of making quick data-driven decisions.

### *A. Machine Learning Algorithms*

All the decisions that a player must take into MOBA videogames are influenced by several variables (e.g., his current position in the map, the position of his opponents and teammates, his virtual avatar and the virtual avatar chosen by the opposite team, etc.). Usually, these decisions offer a great number of possibilities, for instance, a player could buy until six items from a pool higher than three hundred ($6.9 * 10^{14}$ different possibilities). The number of variables to consider and the wide specter of possible solutions make MOBA videogames a multi-dimensional environment appropriate for experimentation.

In consequence, a wide variety of machine learning techniques have been used with different purposes. Most common purposes are to predict an outcome such as whether a team will win a match or if a virtual avatar will die during a fight, to generate a recommendation of an in-game action, and to find the best solution from all feasible options. This section will introduce machine learning techniques grouped by these three purposes: Prediction models, recommendation systems and optimization algorithms.

*1) Prediction Models*

Researchers have studied how to use independent variables during a match such as the virtual avatar chosen by players or the current map position of virtual avatars, to predict an outcome. The review of the papers predicting an outcome generated a list of techniques used. They were grouped based on their similarity to identify the most popular. Fig. 1 shows that most popular techniques are regressions, tree-based models, and neural networks architectures.

As seen in Fig.1 regression techniques are the most used techniques to predict outcomes in MOBA videogames, standing out the Logistic Regression. It is a classic machine learning technique which uses a matrix of predictors ($X$) and a vector of outcomes ($y$), to understand how changes in predictors have impact in the value of the outcome. Logistic regression has been used, mainly, to predict victory [10] [15] [16] [17] [18] [19].

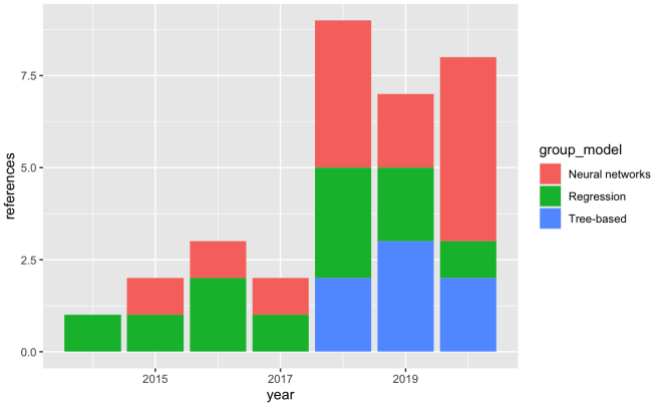


*Fig. 1 Number of reviewed references per year grouped by technique category*

Other popular techniques for predicting outcomes are tree-based techniques. They are usually iterative procedures which build a set of rules to classify each characteristics vector based on its class. During this process, tree-based techniques find how predictors are related to the outcome. One of their main advantages is the creation of oblique separating planes that allow to have a low bias. Papers using tree-based techniques describe applications such as: classification trees to predict whether a MOBA videogame will be successful [20], behavior trees to predict the next action in a game [21] and random forest to predict the game outcome [22] [23]. In the tree-based group, random forest is the only method used more than once.

Thanks to the increasing popularity of neural networks in recent years, it is possible to find research using these techniques to predict some outcomes in MOBA videogames. An Artificial Neural Network (ANN) is a group of interconnections between the neurons of different layers on a system. They are usually made up of three layers: the first layer has input neurons that capture the information of the independent variables (predictors), this information is sent through the synapse to the second layer of neurons, and finally through new synapses it carries the information to the output layer. Recently, researchers have studied more complex architectures making of neural networks into one of the more robust and more popular machine learning techniques.

Artificial Neural Networks have been used to predict which team will win a game. More complex architectures have also been used, for instance, recurrent neural networks to predict a game outcome [24] or the next pick into the pick and bans phase [25]. At last, deep neural networks predicting deaths during a team fight [26] [27].

Finally, other not so used techniques will be mentioned. For instance, predict the winning team using singular value decomposition [28] , factorization machine [16], support vector machines [29], gradient boosting decision [16], several boosting methods [22], naïve Bayes classifier [16], confidence-calibrated model [30], fuzzy logic predicting future states [31], and to predict the picks during the picks and bans phase using Bayesian networks [25].

*2) Recommendation Systems*

Inside a MOBA videogame match, players face real-time situations where they should take decisions into a wide spectrum of possibilities. It could be an overwhelming task specially to novice player. For that reason, researchers have studied how to use machine learning algorithms to give recommendations to players during a match. Reviewed papers are focused on two applications. First, to recommend the virtual avatar to choose at the beginning of the match. Second, to recommend items to buy during a match.

In classic MOBA videogames, two teams of five players face each other. Every player must choose a virtual avatar from a pool higher than 150 different options, each one with different attributes. It means that the final teams' composition is one from more than $4.2 * 10^{21}$ possibilities. In addition, players decision must consider the synergies with the virtual avatar chosen by teammates and weakness of virtual avatars chosen by enemies. These factors stimulate the application of neural networks architectures to suggest a virtual avatar to play during a match. Some used architectures are artificial neural networks [11] [19], recurrent neural networks [32], fully-connected neural networks [33]. As well, researchers have used rule-base systems [34] [35], random forest [18] [33], support vector machines [18], several boosting methods [18] and fuzzy logic [36] methods to recommend the virtual avatar to choose.

On the other hand, during a match each player could use in-game gold to buy a limited number of items which improve the attributes of his virtual avatar. In League of Legend each player could acquire up to six items from a pool higher than three hundred options. In consequence, the possible combinations of items at the end of the match are more than $6.9 * 10^{14}$ possibilities. Some of the techniques used to recommend which item to buy are: clustering methods [1], deep neural networks [37], fully-connected neural networks [37] and Long-short Term Memory networks [38]. Nevertheless, players need a determined amount of in-game gold to buy items during the match. It means that not all the available items are actually feasible options, and this limitation has not been considered in none of the studied papers.

*3) Optimization Algorithms*

An optimization problem is the one that finds the best solution -maximizing or minimizing a specific metric- from all

feasible solutions. It means the optimization problem should consider the mathematic formula to maximize -or minimize- and, also, a group of constraints. These characteristics were found only in one research. It uses Genetic Algorithms to maximize the probability to win a match recommending the virtual avatar to choose. It has as constraint the virtual avatars already chosen by his teammates and enemies during the bans and picks phase [12].

No other paper using optimization techniques was found in the literature review. Even though purchase of items is highly constraint by in-game gold.

In-game gold is acquired with actions through the game, such as, killing neutral monsters, killing enemy minions, destroying structures, etc. In consequence, a player could not be able to buy the 'best' item to maximize his chance to win because he has not enough gold. Then, the possible items to buy should be delimitated to a list of feasible options. Considering the problem of what items to buy as an optimization problem is one of the contributions of this paper to the state of the art.

### *B. Rational Agents*

Machine learning techniques mentioned in previous section could be used as layers into a more complex system which perceives the environment, process information and take a decision. It is known as a rational agent. An entity which perceives its environment, takes actions autonomously in order to achieve goals, and may improve its performance with learning or may use knowledge. In specific, Russel and Norvig [39] describe a goal-based agent as an agent which use goal information to describe situations that are desirable as shown in Fig. 2. It provides the agent a way to choose among multiple possibilities. This section will introduce some applications using machine learning techniques in rational agents applied to MOBA videogames.

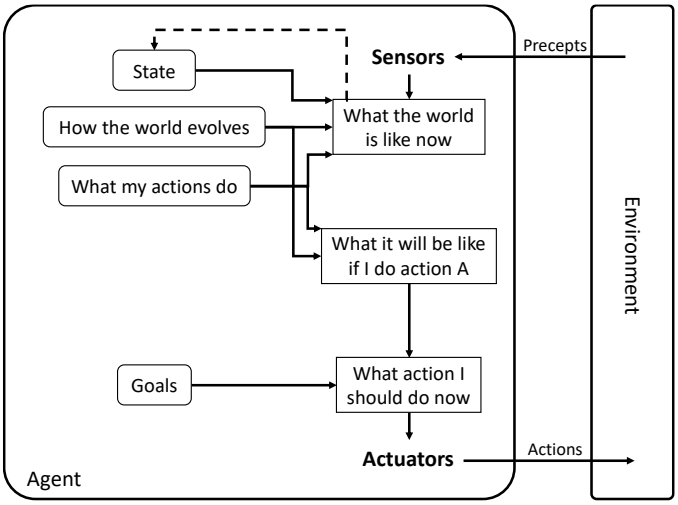


*Fig. 2 Model-based, goal-base. Adapted from Artificial Intelligence: A modern Approach*

Do Nascimento and Chaimowicz [13] describe an Artificial Intelligence capable of playing a MOBA videogame using rule-based systems. They propose the two-layer architecture shown in Fig. 3. The first layer determines the agent's navigation, using influence maps and potential fields techniques. The second layer is used for micromanagement, it determines whether to attack an objective using target selector and an algorithm called orbwalker, to inform when to attack or to move.

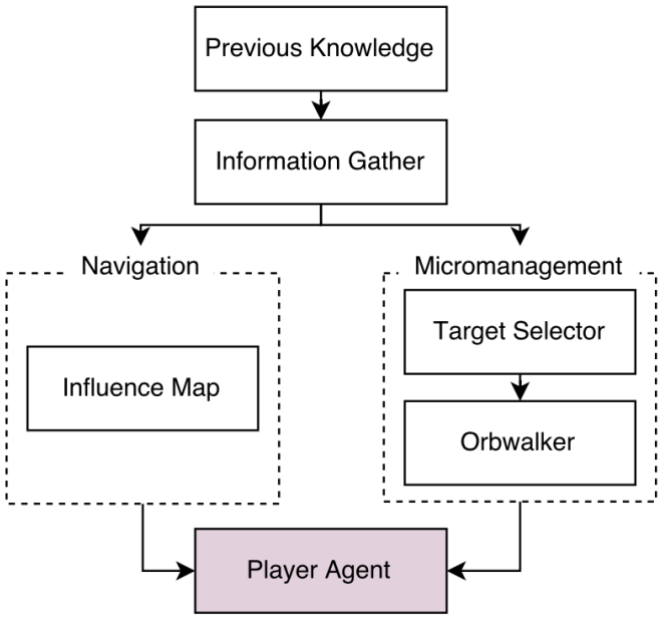


*Fig. 3 Two-layer Architecture for a Rational Agent who plays League of Legends*

Silva, Do Nascimento and Chaimowicz [40] use rational agents to improve the player experience. They use rule-based systems to adjust the difficulty of the game according to player's performance. It uses as sensors the historical player's performance and in-game goals to adjust difficulty into: Easy mode, regular mode, and hard mode. Once the player is classified, the rational agent activates or deactivates the behaviors shown in Fig. 4.

| | | Artificial Intelligence | | |
|---|---|---|---|---|
| | | Easy Mode | Regular Mode | Hard Mode |
| Defense Strategy | Defend Allied Towers | X | X | X |
| | Retreat | X | X | X |
| | Item Manipulation | | X | X |
| Attack Strategy | Main Attack | X | X | X |
| | Target Selection | X | X | X |
| | Track Enemy Hero | | | X |
| | Cast Spell | | | X |

*Fig. 4 List of features to activate or deactivate based on dynamic difficulty*

Pratama, Nugroho and Yuniarno [41] described a rational agent which adjusts the difficulty of the videogame. Their rational agent uses several in-game metrics to decide the bot's difficulty with fuzzy distributions. Fuzzy distributions calculate a capability score from 0 to 100 and assign a label from: Not Capable, Somewhat Capable, Capable and Very Capable. Proposed architecture could be observed in Fig. 5.

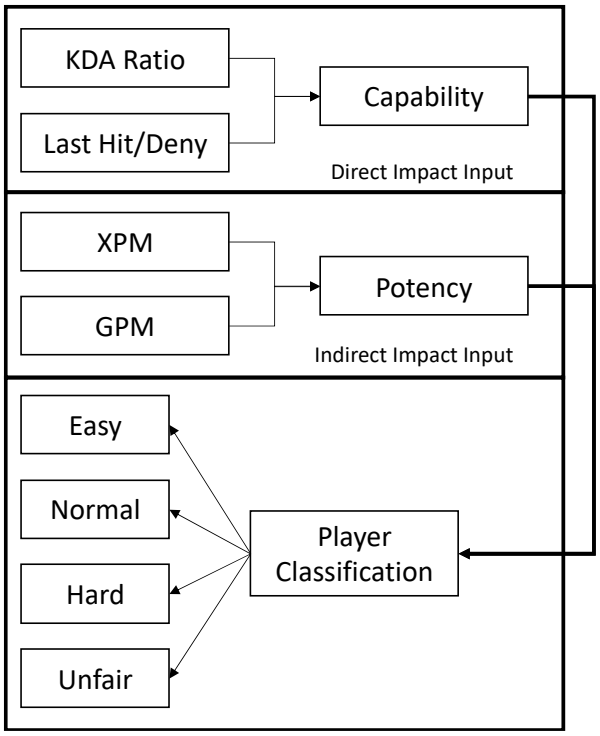


*Fig. 5 Dynamic Difficulty Adjustment using Fuzzy-controller Rational Agent. Adapted from Fuzzy controller based AI for Dynamic Difficulty Adjustmentfor Defense of the Ancient 2 (DotA 2)*

Do Nascimento and Chaimowicz [42] also proposed a rational agent who serves as a tutor for novice players. It detects the specific player's situation, goes through the behaviors tree, and takes actions. The Fig. 6 shows the behaviors tree proposed using arrow rectangles to represent sequencers, question circles to represent selectors and rectangles to represent actions.

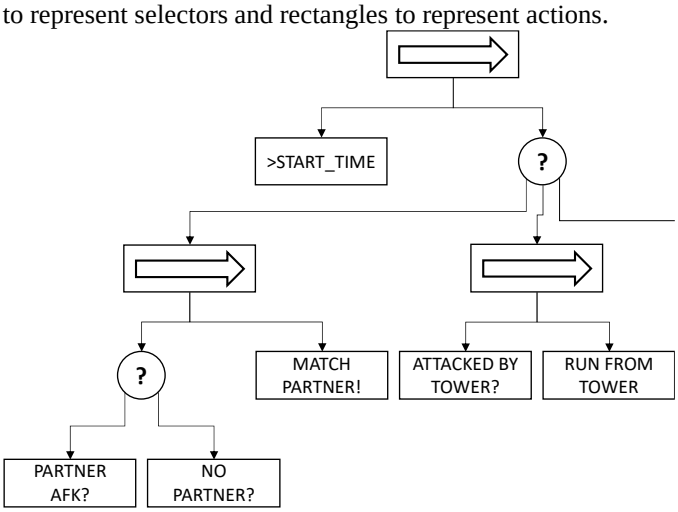


*Fig. 6 Partial Behavior Tree implemented by Rational Agent. Adapted from A tutor agent for moba games.*

Next section shows the insights obtained after an extensive literature review and analysis.

### *C. Summary*

Based on previous works, described above, it is possible to identify:

- Researchers have used various machine learning techniques in the MOBA videogames context to predict several outcomes, recommend the next best action during a match, and optimize the virtual avatar to choose.
- Most popular techniques for prediction are: Logistic Regression, Support Vector Machines, Artificial Neural Networks and Random Forest.
- Research using recommendation systems have focused their efforts in two applications: To recommend the virtual avatar to choose at the beginning of a match, and to recommend the items to buy during a match.
- None of the reviewed papers using recommendation systems to decide what items to buy, consider the in-game gold as a constraint.
- Optimization techniques have been the less explored techniques in MOBA Videogames.
- Machine learning techniques have been used in different layers of rational agents.

Considering the listed insights, the next section will describe the proposed methods to design a rational agent architecture. The agent will use machine learning techniques to perceives the current in-game situation and suggest what items to buy during a game in order to maximize the chance to win a match, taking in consideration the available in-game gold of the player as a constraint. In consequence, the item recommendation will be gotten using an optimization algorithm.

## III. METHODS

Inspired in the goal-based rational agent architecture shown in Fig. 2, the Fig. 7 introduces the proposed rational agent architecture for this investigation. The agent processes were split in offline and online processes. For that reason, agent has two sensors that help the agent to perceive its environment: historical and real-time. They will be described in detail in Section A.

The information gotten by the sensors is filtered and transformed into a structured form. It is the state of the environment. This is also useful for representing possible future states. Then, Section B details which relevant variables are included into the state and their format.

As the rational agent is a goal-based agent, it needs a component which understand the state in terms of its goal. The agent's goal is to maximize the probability of the player to win the match. For that reason, Section C describes the component of the agent that translates the state into the probability of winning the match.

Finally, using the current state and possible future states, the agent uses an optimization algorithm to generate a recommendation. Thereby, Section D gives details about the component of the rational agent that makes the decision about which item to buy to maximize the odds of winning a game.

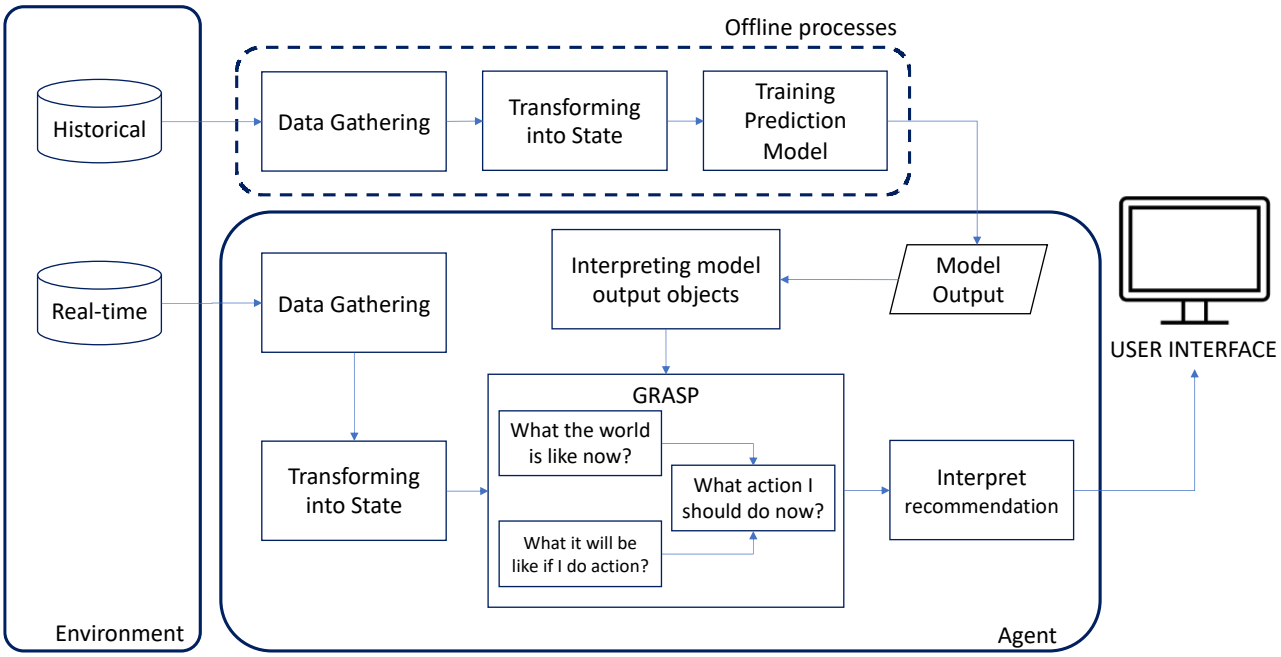


*Fig. 7 Proposed rational agent architecture.*

The online processes of the agent allow to generate recommendations almost in real-time. These recommendations will be different each time the current state of the match change, then, the recommendation is dynamic and in accordance with the situation of the game.

It is important to mention that the proposed rational agent architecture in this investigation does not have actuators. Because it delivers the information to the player. Is the player and not the agent, who transforms the environment following - or not- the recommendation of the agent.

### *A. Data Gathering*

In this architecture, two different sensors are needed. Historical sensors which will provide the required information to train machine learning algorithms that help the agent to interpret the environment, and the impact of its actions. And real-time sensors which will be used to take decisions during a match.

#### *1) Historical Sensors*

RIOT games, the developing company who creates League of Legends, made available to the public an API where information related to the game could be downloaded [43]. It is mainly used for developers to create tools which improve the user experience, or data analysts hired by professional teams to design strategies and help professional players to increase their performance.

For this investigation, historical information was extracted from three different endpoints: Champion [44], Match-v5 and Item [45]. The obtained dataset was compared against four large datasets available to the public in Kaggle webpage. The four datasets chosen as benchmark will be called in this paper accordingly to the user who published it: Mitchell J [46], Chuck Ephron [47], mitchel's fanboi [48] and Paolo Campanelli [49]. They were compared based on the size and the diversity of the dataset.

A total of 554.671 matches were downloaded from the RIOT Games API. Fig. 8 shows that the obtained dataset using the League of Legends API contains at least 3x more observations than benchmark datasets.

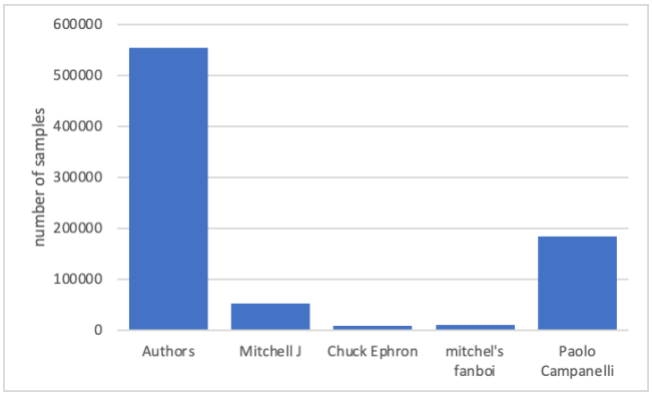


*Fig. 8 Number of Samples of Obtained Dataset versus Number of Samples in Benchmark Datasets.*

As League of Legends is a worldwide played videogame, it has players across the globe. They have access to servers located in different locations to improve their connections to the videogame. It implies the information is split in several servers, then, to train a rational agent who is not biased by regional behaviors is necessary to gather information from each server.

Benchmark datasets only have information of one server: Mitchell J [46] has information from West Europe (EUW1) exclusively, Chuck Ephron [47] has information from competitive matches, mitchel's fanboi [48] does not specify server, and Paolo Campanelli [49] has mainly information from EUW1.

Meanwhile, obtained dataset includes around forty thousand matches from each one of the eleven servers around the world - Brazil (BR1), North Europe (EUN1), West Europe (EUW1),

Japan (JP1), Korea (KR), North Latin America (LA1), South Latin America (LA2), North America (NA1), Oceania (OC1), Russia (RU) and Turkey (TR1)-. Fig. 9 shows the number of matches gathered for each server.

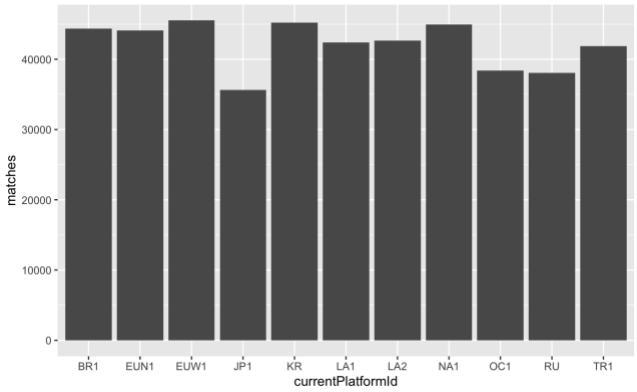


*Fig. 9. Number of Matches Gathered in Each Server*

Inside the servers, the players are grouped based on their performance using an ELO system. The ELO in League of Legends categorizes players in tiers. There are nine different tiers -Iron, Bronze, Silver, Gold, Platinum, Diamond, Master, Grandmaster and Challenger-. In consequence, players are paired in matches with allies and enemies from the same tier.

Reviewed papers used information almost exclusively from high tiers. Benchmark datasets have a similar behavior: Mitchell J [46] does not specify the tier, Chuck Ephron [47] is focused on Challenger, mitchel's fanboi [48] uses Diamond matches only, and Paolo Campanelli [49] does not specify tier neither. Contrastingly, the obtained dataset has an almost balanced number of matches in each tier. In consequence, obtained dataset proved to be better than benchmark datasets in terms of size and diversity.

However, despite its performance against benchmark datasets, the obtained dataset includes some matches that should be removed. Fig. 10 shows a histogram of the duration in seconds of collected matches. Matches with a duration shorter than 540 seconds are cancelled matches because one or more players are absent, then, they cannot be considered relevant. For that reason, they will be removed from the obtained dataset. As result, final dataset contains information from 540.155 different matches from the 554.671 in the original dataset.

Each one of the collected matches in the final dataset generated two characteristic vectors. The first one represents the blue team perspective, then, the outcome associated to this vector is whether the blue team win the match. The second characteristic vector represents the red team perspective and accordingly the outcome associated to this vector is whether the red team win the match.

In consequence, the dataset with historical information used to train the offline process in the rational agent have a total of 1.080.310 characteristic vectors and outcomes. They were split into a training dataset of 629.224 observations, a testing dataset of 269.570 observations, and a validation dataset of 181.516.

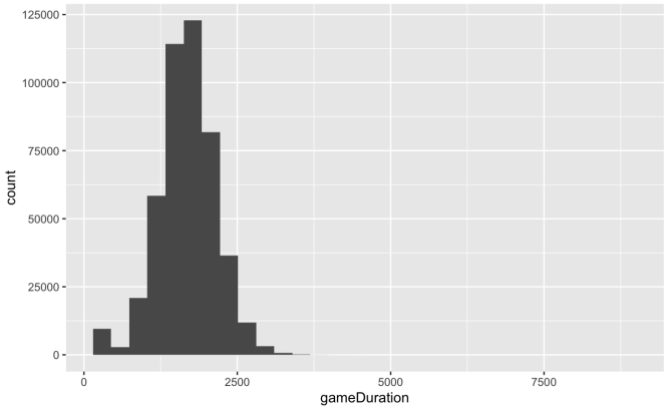


*Fig. 10 Matches Duration in Seconds from All the Collected Matches.*

In conclusion, the diversity, and the size of the obtained dataset -compared against various benchmark datasets- should be enough to train robust machine learning algorithms that will be described in further sections.

*2) Real-time Sensors*

While historical information is used for training the machine learning algorithms in offline processes, the real-time information will be used to perceives the agent's environment in online processes. Real-time information of an active match could be obtained from an API called Live Client Data API. It has been developed also by RIOT games and could be accessed by native applications via HTTPS using Swagger v2 or OpenAPI v3 specs. From this API the used endpoints are Active player, Events and Game.

### *B. State Representation*

Information captured by historical and real-time sensors, should be transformed into a characteristic vector that could be used by the agent as a structured representation of the current state of its environment and further, as an interpretation of the impact of its actions in the state of the match. Therefore, ths information included in the characteristics vector is expected to be relevant. For that reason, literature was reviewed searching for the most representative variables in MOBA videogames to train machine learning algorithms [50].

First, papers using machine learning algorithms in MOBA videogames were grouped into six categories based on questions the research tries to answer: Spatio-temporal, Players, Champions, Team composition, Items and Victory. Second, most used variables in each category were tabulated. Third, the most common variables across categories will be considered as relevant. Finally, once the most relevant variables were identified this paper presents a discussion about the best way to represent these variables.

*1) Spatio-Temporal*

Typically, a MOBA match occurs into a map where two teams appear from different corners of a map connected by three main paths known as Bottom, Middle and Top Lanes. There are also several secondary paths through a neutral territory known as jungle. Fig. 11 shows a generic MOBA map. According with his position the player would face different situations: In main paths players would fight with enemy minions and turrets, in the jungle they could fight against

neutral monsters, and there are neutral objectives more important than others. It suggests the position of players into the map -coordinates- are relevant for the development of the match.

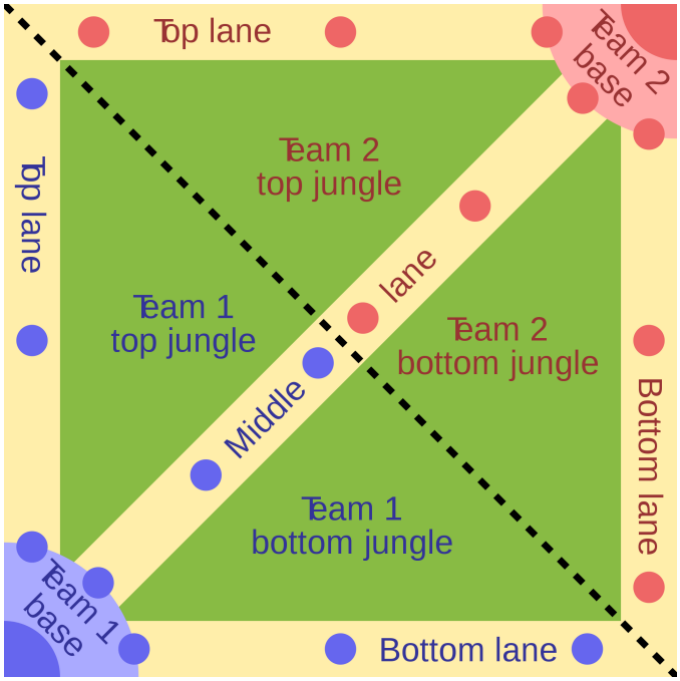


*Fig. 11 MOBA Videogames Generic Map. Yellow lines are the main paths, small circles represent turrets, defensive building between team bases and big circles are the nexus; building that should be destroyed to win the match*

Taking into the account the mentioned importance of the place and the time where in-game actions were performed, researchers have sought patterns related to how the game is developed in different map coordinates and time slices. Some of the spatio-temporal investigation objectives found in the literature are:

- To detect an experienced player based in the way that he navigates across the game map [51] [52] [53] [29].
- To develop rational agents which determine the best way to navigate the game map at a specific moment of the game [42] [13] [37].
- To analyze the advantage of one of the teams at a specific time during the game [54] and specifically, when this advantage generates an irreversible snowball effect [55] [56].
- To improve spectators experience by determining in which sectors of the game map the events most relevant to the game are taking place [57] [58].
- To predict the outcome of an encounter with two or more players [15] [59] [27].
- To determine whether an ability or an actionable item could be used [60].

Even when all these papers are related to spatio-temporal research, the variables used as predictors are very wide. Despite 64 different predictors were identified, only 3 of them are used in more than one of these research.

*2) Players*

MOBA versatility allows every player to behave different to the same situation. For instance, the same virtual avatar could be equipped with different items relaying on the player preferences.

In consequence, this category groups research that analyze characteristics of the players. These characteristics do not depend on the game condition, on the contrary, they tend to be present into every player's game. In this category we found objectives such as:

- To automatically adjust the level of difficulty for any player based on his historical performance [36].
- To analyze the outcome of a game based on the historical performance of the players [61] [24] [62]
- To categorizing players according to their historical performance and design analysis strategies to improve it [63] [53] [64] [29] [65].
- To analyze the learning speed of a new player [17].
- To predict based on their gaming habits if a player is at risk of quitting the game [66], the best way to match players based on their historical performance [26] [67]
- How the player's social network within the video game affects the outcome of his games [68].

As described in spatio-temporal category. The variables used as predictors in these papers are very diverse. In total, they use 51 different predictors. However, only 10 of them are used in more than one research.

*3) Champions*

At the beginning of every game each player must choose a virtual avatar known as champion in League of Legends. This decision should consider the role that the player will perform (e.g., top, mid, jungle), the virtual avatar chosen by enemies and allies, and specially the synergies between them. Then, each team pick five different champions from more than 150 different options.

Due its complexity and importance, the papers grouped in this category have studied the impact of the avatar selected by the players. Mainly these research are focused in designing champion recommendation systems [25] [69] [34] [35] [70] [18] [38] [28] [33] [32] and predict the outcome of a game based on the combination of the avatars choose by the player, his teammates and opponents [16] [71] [11].

Although these investigations use a total of 18 different independent variables, the only one that appears in all research is 'chosen-hero': a binary vector that represents the champion selected by one or more players.

*4) Team Composition*

A limitation commonly mentioned in research is how to introduce into machine learning models the synergies that must exist between players of the same team. Related to this question, the Team Composition category was created for those papers that seek objectives related to the roles of each player within a team [10] [62], the analysis of player behavior in the composition of a team [11] [71] [72] [64] [73], predicting the outcome of a game considering synergies between players [16] [74] or optimizing the construction of a team of players [34] [32] [12] [75] [67].

A total of 45 different independent variables were identified in the investigations within this category. Only 4 of them were

used on more than one paper.

*5) Items*

During each game every player has the possibility to acquire items which improves the attributes of his virtual avatar. These items are bought using in-game money that the player can get performing several actions (e.g., Killing neutral characters). In League of Legends, throughout a match each player has the possibility of acquiring up to six items from a pool of almost 300 different objects, each with different benefits for the character.

The research that study the impact of items into the match were grouped within this category. They sought to develop systems that recommend to the player which are the items to acquire during the game [1] [76] [19] [37].

In a similar way as in the champions category, the only variable that appears more than once in papers into the item's category is 'items': a binary vector that identifies whether a champion has a specific item equipped.

*6) Victory*

Another of the objectives found in literature, is the prediction of the winning team in a game. Papers into Victory category use variables known at the beginning of the game [16] [69] [28] [29] [24] [74] [22] [23], at any time during the game [15] [61] or variables known at the end of a game in order to understand their influence on the result [77] [30] [78].

Of the 49 independent variables used within this category, only 9 of them have been used on more than one research.

*7) Summary*

Finally, as mentioned during the categories description it is evident that the used predictors are very heterogenous, even within the categories. A total number of 163 predictors were identified in the literature. 32 of them were used on more than one paper. Table 1 shows predictors used on five or more different papers.

TABLE 1
MOST FREQUENTLY USED PREDICTORS

| Variable | Number of Appearances |
|---|---|
| Chosen hero | 11 |
| Gold | 9 |
| Kills | 8 |
| Deaths | 6 |
| Creeps | 6 |
| Gold difference between teams | 6 |
| Roles (e.g., carry, mid-lane solo) | 5 |
| Assists | 5 |
| Items | 5 |

In addition to variables most frequently used, the variables used crosswise in various categories were also taken into the account. Fig. 12 shows the interception of variables in three categories: Players, Champions, and Items. The unique variable used on the three categories is the chosen hero, at the same time, items and roles are used on two of the three categories with common variables.

Then, since they are the only variables used crosswise in papers with different investigation objectives, these three variables -chosen hero, items, and roles- will be considered relevant variables in MOBA videogames and, in consequence, used as predictors in this investigation.

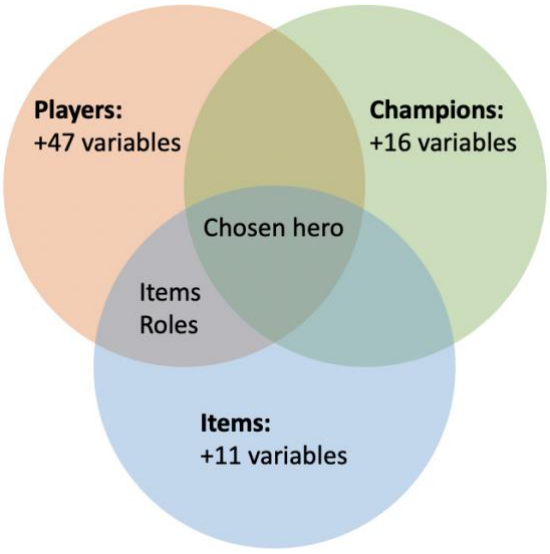


*Fig. 12 Variables Interception Across Several Goal Categories.*

It is important to mention that items and chosen hero variables have been commonly used in the literature as a Boolean vector. The vector position which represents a champion is 1 when the champion has been chosen and 0 in other cases. Same with items, the vector position for each item is 1 once the item has been bought and 0 in other cases.

However, this vector representation has two analytical difficulties to take into consideration. First, due the number of existing items and champions, each player creates a vector with more than 700 characteristics. Just seven of these characteristics have a value of 1 -six bought items and one chosen champion- and the rest of them are 0. Therefore, it creates a matrix with a lot of dispersion due the number of zeros. Second, attributes of champions and items change almost twice a month because developers apply new patches to fix bugs and to balance the victory ratio of each virtual avatar [79].

Both, disperse matrix and changes in attributes, would have an impact in the trained models making our rational agent difficult to sustain, especially, when a new patch is applied.

For that reason, in this investigation is proposed a different interpretation for champions and items variables. It is the translation of the selected champion and bought items into their attributes. In League of Legends there are thirteen attributes: attack damage, attack speed, critical hit chance, life steal percentage, hit points, spell block, armor, mana points, magic damage, magic damage, movement speed, hit point regeneration, attack range and mana point regeneration.

The proposed representation allows to represent the state of the game with a vector of one hundred thirty characteristics -thirteen attributes for ten players-, without zeroes. The specific state of a player -the champion he chosen and the items he bought- is represented by his own thirteen variables. It gives to our rational agent the capacity to provide different recommendation to each one of the players involve into a match, and different recommendations during the game based on the level of their champions and the items that they have

already bought.

### C. State Interpretation

Once the agent receives the characteristics vector described in Section B it needs a component to interpret its position on the goal, to winning the match. This component should be able to answer the questions: What are my probabilities to win the match based on the current state? And, how much my probability of win the match increases after buying an item?

For this component the method selected was a prediction model that uses the characteristics vector to predict the chance to win a match. Based on the literature review, the most popular machine learning techniques to predict outcomes in MOBA videogames are: Artificial Neural Networks (ANN), Logistic Regression (GLM), Random Forest (RF) and Support Vector Machines (SVM).

Those four techniques were used for experimentation. The most suitable technique was selected based on four different metrics: Accuracy, Stability, Training time and Complexity.

#### 1) Accuracy

To predict whether a team will win a match is a classification problem, due the outcome takes just discrete values: victory or defeat. A commonly used metric in classification problems is the accuracy. It measures the percentage of observations whose actual class matches with the class predicted by the machine learning algorithm. If we consider victory is the true class, then accuracy could be expressed as follows.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Where, true positive (TP) is the number of observations classified by the algorithm as victory, which are real victories. True negative (TN) is the number of observations classified by the algorithm as defeat, which are real defeats. False positive (FP) is the number of observations classified by the algorithm as victory, which are actually defeats. Lastly, False negative (FN) is the number of observations classified by the algorithm as negative, which are actually victories.

#### 2) Stability

This metric evaluates the consistency of the accuracy obtained by each algorithm with different size in the training dataset. Stability is measured using the relative standard deviation of the accuracy obtained during every experiment performed, grouped by prediction technique.

#### 3) Training time

It is the time in seconds that each algorithm takes to be trained. Comparison between techniques is done using is the time obtained in the experiment with the greatest training dataset.

#### 4) Complexity

This is the last metric to evaluate, it is a qualitative evaluation used by the authors to evaluate how simple could be implement the information provided by the prediction model into an optimization algorithm. Each technique will get a qualitative score based on the output of the algorithm, and the operations the agent should perform to transform a characteristic vector into the probability to win a match.

#### 5) Model Selection

Performance metrics will be tabulated and normalized inspired by TOPSIS methodology [80]. It is commonly used in multi-objective optimization problems where variables to maximize or minimize have different scales. After normalized the metrics they can be weighted to calculate an overall score.

### D. What Action Should I Do Now?

Setting, the optimization problem where the rational agent selects what items to buy maximizing its probabilities to win from the items that could be bought with its available in-game gold, could be analyzed as a knapsack problem. It is a problem in combinatorial optimization often solved with greedy algorithms, in consequence, this investigation uses GRASP as the technique which will help the agent to take final decision.

#### 1) Knapsack Problem

Richard Karp describes knapsack problem for the first time [81] as a NP-complete problem, where: There are $n$ different items that could be introduced into the knapsack with an inventory of $q_i$. Also, every item $i$ has a benefit $v_i$ and a weight $w_i$. At the same time, the knapsack has a finite capacity $W$. Then, mathematical problem could be expressed as:

$$maximize \sum_{i=1}^{n} v_i x_i$$

Subject to:

$$\sum_{i=1}^{n} w_i x_i \leq W$$
$$0 \leq x_i \leq q_i$$
$$\sum_{i=1}^{n} x_i \leq 6$$

For the problem in this paper: $v_i$ represents how much increase or decrease the probability to win a match when player buy the item $i$, the decision variable $x_i$ is the units of the item $i$ that player buys, $w_i$ is the in-game gold price of the item $i$, $W$ represents the amount of available in-game gold, and $q_i$ will be unbounded for almost every item.

Additionally, based on League of Legends rules more constraints are required: The sum of all $x_i$ must be lower or equal to six, and some items are mutually exclusive (e.g., Berserker's Greaves and Mercury's Treads).

Finally, as items have not the same utility with different champions a factor $s_i$ will be included in objective function. Where, $s_i$ is the cosine similarity between the current state and the state after the player acquires an item. For instance, if a champion has a high attack damage and a low magic damage, the rational agent will not suggest an item with high magic damage and low attack damage. It happens because $s_i$ will reduce the utility of the item, due the dissimilarity between current and future state. Then, the optimization problem is:

$$maximize \sum_{i=1}^{n} s_i v_i x_i$$

*2) Greedy Randomized Adaptive Search Procedure (GRASP)*

This metaheuristic introduced in 1989 is commonly used to solve problems in combinatorial optimization [82]. It works in two stages. First stage builds random solutions, and second stage improves this solutions through local searches.

```
procedure GRASP(Max_Iterations, Seed)
1   Read_Input();
2   for k = 1,..., Max_Iterations do
3     Solution ← Greedy_Randomized_Construction(Seed);
4     Solution ← Local_Search(Solution);
5     Update_Solution(Solution, Best_Solution);
6   end;
7   return Best_Solution;
end GRASP.
```

In first stage, items are randomly selected from a Restricted Candidate List (RCL). They are included into the initial solution whether they contribute to objective function.

```
procedure Greedy_Randomized_Construction(Seed)
1   Solution ← ∅;
2   Evaluate the incremental costs of the candidate elements;
3   while Solution is not a complete solution do
4       Build the restricted candidate list (RCL);
5       Select an element s from the RCL at random;
6       Solution ← Solution ∪{s};
7       Reevaluate the incremental costs;
8   end;
9   return Solution;
end Greedy_Randomized_Construction.
```

Elements in the RCL are filtered using an $\alpha$ parameter settled by user. Formulas to build the RCL for maximization problems where $f_u(e_i)$ is the incremental benefit of the item $i$, are described below.

$$lb = \min_{\forall i \in Solution} \{f_u(e_i)\}$$
$$ub = \max_{\forall i \in Solution} \{f_u(e_i)\}$$
$$R = ub - lb \quad 0 \leq \alpha \leq 1$$
$$RCL = \{e_i | lb + (1-\alpha) * R \leq f_u(e_i) \leq ub\}$$

Values of $f_u(e_i)$ are provided by the prediction algorithm described in Section C. Furthermore, second stage improves initial solution using local search operators.

```
procedure Local_Search(Solution)
1   while Solution is not locally optimal do
2       Find s' ∈ N(Solution) with f(s') < f(Solution);
3       Solution ← s';
4   end;
5   return Solution;
end Local_Search.
```

As player can sell a previously purchased item, the suggested local search operator is to change one of the items into the initial solution with the highest utility item that has not been included yet.

## IV. Experimentation & Results

Experimentation was designed based on what a player could need from the rational agent. Players need the agent makes a proper interpretation of the state of the game. They also need that recommended items really bring them closer to their goal, and that the recommendation is delivered in time -before the state of the game changes-.

Based on these need the experimentation was designed. Then, Section A describes a comparison between machine learning techniques commonly used for prediction to decide which algorithm to use into the offline processes of the agent. Section B describes the experiments performed to test the optimization algorithm into the online processes of the agent.

### *A. Evaluation of Machine Learning Techniques*

Thanks to the literature review was possible to determine the most popular machine learning techniques for prediction in MOBA videogames are: Logistic Regression (GLM), Random Forest (RF), Support Vector Machines (SVM) and Artificial Neural Networks (ANN). These techniques were analyzed following four different criteria: accuracy, stability, training time and complexity.

The four techniques were trained several times with different samples of the training datasets. The sample size varies in each iteration from 31.371 to 641.805, growing steadily. Each time one of the techniques were trained, the accuracy and training time was measured.

*1) Accuracy and Stability*

Fig. 13 shows the accuracy obtained in each experiment. Horizontal axis is the size of the dataset used in the experiment and vertical axis is the obtained accuracy.


*Fig. 13 Accuracy Obtained by Each Machine Learning Algorithm with Various Sample Sizes.*

The technique with the lowest accuracy is ANN. It achieves a maximum accuracy of 0.8212. Also, its results are not stable. Its accuracy in the smallest datasets is lower than 0.80. RF has also low accuracy. Its accuracy in the experiment using the whole dataset is 0.8391. RFF accuracy is stable; however, it shows a decreasing trend, with biggest datasets its accuracy drops slightly. Meanwhile, SVM has in almost every

experiment an accuracy around 0.875. Its final accuracy is 0.8797.

In terms of accuracy and stability the technique with best performance is GLM. Most experiments with GLM have an accuracy higher than 0.90. The experiment with the biggest dataset using GLM gets an accuracy of 0.9012.

However, accuracy is lower than expected. For that reason, further explorations should be done to understand: Why is the best accuracy just 0.90? and, is there any pattern into the matches affecting the metric? Two additional analyses were done to answer these questions. These analyses use results of GLM, because it is the technique with highest accuracy.

First, matches in testing dataset were split by region. The accuracy of the model was evaluated in each region to understand if there is a regional pattern affecting the global accuracy. Fig. 14 shows the overall accuracy by region. The accuracy obtained in every region is similar, then, it is possible to conclude that there is no regional pattern that impacts global accuracy.

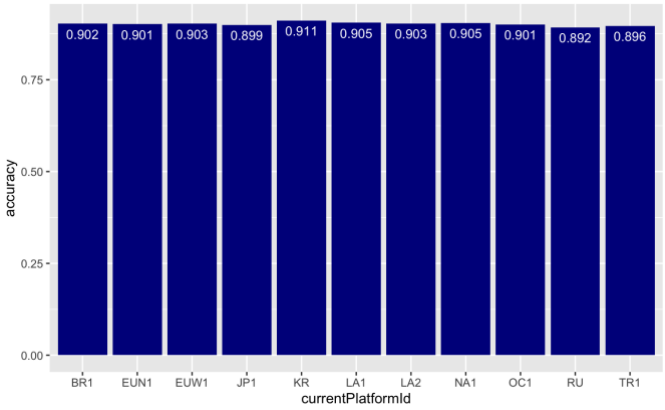


*Fig. 14 Accuracy Obtained in Testing Dataset by Region with Logistic Regression.*

Second, matches were grouped accordingly with their duration. Fig. 15 shows the accuracy by duration. It is possible to identify a pattern. GLM has an accuracy higher than 0.95 in matches with a duration lower than 28 minutes. From minute 28 to 30 the accuracy drops; however, it remains over 0.90. After, in matches with a duration higher than 30 minutes the GLM accuracy decreases constantly.

A hypothesis about this behavior could be found in what papers call 'late game'. After the minute 25 almost all the champions have reached their maximum level and has bought their maximum number of items. In consequence, from a quantitative perspective the game starts to balance. For that reason, during the late game quantitative attributes give importance to qualitative aspects such as strategic decisions or synergy between players -different from the synergy between virtual avatars-.

A rational agent which suggests what items to buy is useful for the player before he buys all his items. It means, the accuracy of the prediction model is not as relevant during late game when he has bought his maximum number of items. Nevertheless, during the early stages of the game is when the agent can produce a real advantage for the player. During early stages GLM has an accuracy around 0.975.

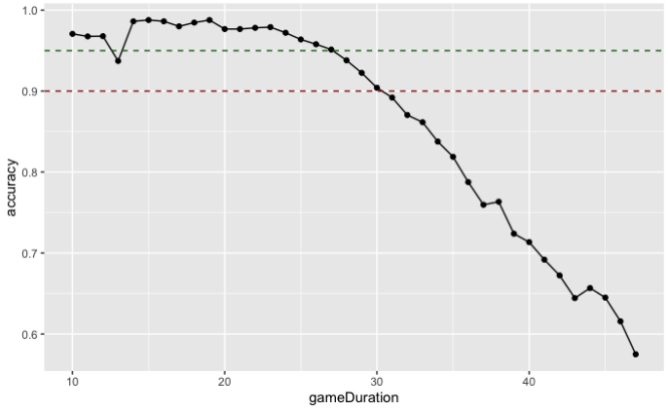


*Fig. 15 Accuracy Obtained Based on Game Duration with Logistic Regression.*

*2) Training Time*

For each experiment, the time in seconds that the algorithm took to be trained was measured. Due the size of the dataset, these experiments were executed using Databricks in a standard cluster with a Runtime 8.2 configuration [83]. Fig. 16 shows a comparison of the training time in each experiment.


*Fig. 16 Training Time in Seconds by Each Machine Learning Algorithm with Various Sample Sizes*

In every experiment, RF was the techniques with the higher training time. It scales steadily until reaches 492 seconds in the experiment with the biggest training dataset. ANN increase constantly as well. It reaches a training time of 152 seconds. Meanwhile, SVM and GLM training time remains stable during the experiments. Size of dataset does not have a great impact in the time these techniques need to be trained. SVM requires 63 seconds to be trained with the biggest dataset. While GLM is the techniques with the lowest training time. It requires only 51 seconds to be trained.

*3) Complexity*

As mentioned in Methods, it is a qualitative evaluation related to the complexity of the output of each machine learning technique. Each algorithm was evaluated as Low, Medium, and Hard.

RF is an ensemble learning method. For that reason, the

output is a group of weak predictors, specifically 300 classification trees. Then, it is not possible to determine the impact of purchasing an item on the probability of winning the game without going through all the trees. Consequently, the complexity of RF is considered high.

ANN output is a group of four coefficient matrixes and four intercept vectors, due the trained ANN has four layers. Within this structure is difficult to isolate the benefit of buying an item in the probability to win the match. In consequence, the complexity of this model is considered medium.

SVM generates a group of vectors which represent the hyperplanes that separate classes, also, a vector of weights with a length equal to the number of separating hyperplanes. Despite the operations required to transform the state into the chance to win a match are simple, is difficult to isolate the impact to purchasing an item into the probability of winning. Accordingly, SVM complexity is considered as medium.

Finally, GLM complexity is considered as low. GLM outputs are the simplest. They are a coefficients vector and an intercept. The coefficients could be easily interpreted as the impact of each item facilitating the process into the optimization algorithm.

*4) Models Comparison*

Table 2 shows the comparison between selected machine learning algorithms. It summarizes the accuracy, stability, training time and complexity described in previous sections.

TABLE 2
MACHINE LEARNING ALGORITHMS COMPARISON

| Metric | ANN | GLM | RF | SVM |
|---|---|---|---|---|
| Accuracy | 0.8212 | 0.9012 | 0.8391 | 0.8797 |
| Stability | 0.0257 | 0.0003 | 0.0029 | 0.0060 |
| Time | 152.48 | 51.19 | 492.58 | 62.71 |
| Complexity | Medium | Low | High | Medium |

Nevertheless, the four metrics could not be weighted because their dimensions are different. Then, for Table 3 the metrics were normalized using the next steps:

1. To identify the highest score in each metric.
2. To divide each result into the best score, identified in previous step. It transforms every variable into a range from 0 to 1.
3. If metric is better when is lower, then, result of previous step were subtracted from one.

TABLE 3
NORMALIZED MACHINE LEARNING ALGORITHMS COMPARISON

| Metric | ANN | GLM | RF | SVM |
|---|---|---|---|---|
| Accuracy | 0.9112 | 1.0000 | 0.9311 | 0.9762 |
| Stability | 0.0000 | 0.9866 | 0.8851 | 0.7642 |
| Time | 0.6904 | 0.8961 | 0.0000 | 0.8727 |
| Complexity | 0.5000 | 1.0000 | 0.0000 | 0.5000 |

As metrics are in the same range they could be weighted into an overall score. The four metrics will be equally averaged, the result could be observed in Table 4.

Based on numbers in Table 4 it is possible to conclude that a rational agent which uses GLM as prediction model in its offline processes performs better than others. This agent would give to the player more certainty about its prediction -accuracy-, with more consistency -stability-, and would take less time to be re-trained for maintenance -training time-. The second-best rational agent would be the one which uses SVM.

TABLE 4
PREDICTION MODELS SCORE

| Metric | ANN | GLM | RF | SVM |
|---|---|---|---|---|
| Final Score | 0.5254 | 0.9707 | 0.4541 | 0.7783 |

## B. Evaluation of Optimization Algorithm

This section describes the experiments performed to measure the optimization algorithm into the online processes of the rational agent. One thousand real states were randomly selected from Live Client Data API. For these states, recommendations were generated using the optimization algorithm. The RCL for each experimented were created with three different values of alpha (30%, 50% and 70%).

Optimization algorithm must deliver recommendations in near real-time so that the player purchases the items before the state of the game changes. Also, the recommendation must increase the probability of the player of winning the match. Consequently, in each experiment were measured the time that the algorithm takes to generate a recommendation -execution time- and the increment in the odds of winning the game -odds incremental-.

*1) Execution Time*

This metric was measured taking the time from the moment when characteristics vector is delivered to GRASP algorithm, to the moment when optimization algorithm delivers a recommendation. The recommendation could include one or several items to buy or to sell.

The average execution time on the experiments is 0.3556 seconds. Fig. 17 shows that execution time is independent to alpha values. There is no significant difference in execution time distributions, then, does not matter the alpha value to use in the rational agent because that will respond at the same speed.

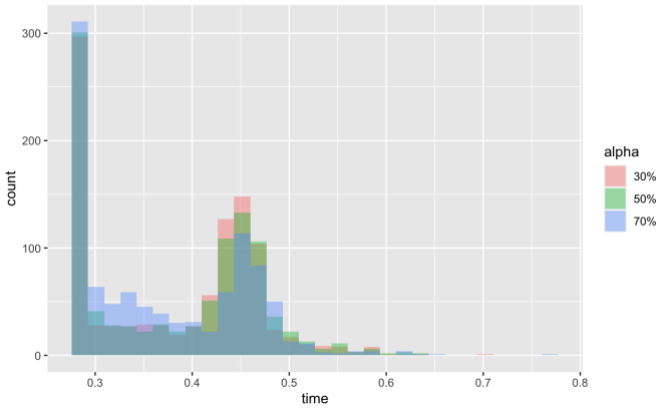


*Fig. 17 Execution Time Distribution Grouped by Alpha Value*

*2) Odds Incremental*

The rational agent can translate the state of the game into the probability of winning a match. In consequence, it can translate simulated state (e.g., what will be the state of the game if the

player buys the recommended item?) into the probability of winning a match. Probabilities obtained of current state and simulated state, are used to calculate the odds. In this paper, the difference between the odd of simulated state and the odd of the current state is called Odds incremental.

Odds incremental results could be observed in Fig. 18. Experiments with 70% of alpha have an average Odds incremental of 5.2213, experiments with 50% of alpha have an average Odds incremental average of 5.1616, and experiments with 30% of alpha value have an average Odds incremental of 5.5496. In addition, Odds incremental with 30% of alpha are most consistently focus around the mean. The Odds incremental standard deviation with 70% alpha is 2.2664, meanwhile, the Odds incremental standard deviation with 30% alpha is 1.4205.

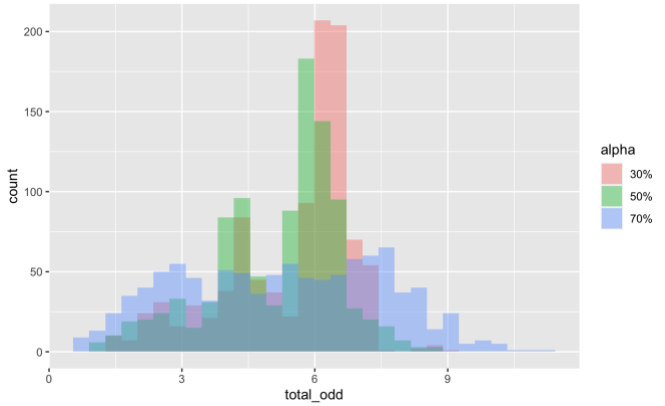


*Fig. 18 Odds Incremental Distribution Grouped by Aplha Value*

Higher dispersion in experiments with higher alpha values, happens because with higher alpha values the RCL includes items with lower cost/benefit ratios. While the lowest alpha value shows more consistent results around the average.

Through a hypothesis test is possible to determine that: Results using a 30% alpha value are at least 0.3146 higher than results with 50% alpha value. And at least 0.1769 higher than results obtained with 70% alpha value.

However, described behavior is different when player has available a higher amount of in-game gold. It could be seen in Fig. 19. 70% alpha value gets higher odds incremental with greater amounts of in-game gold.

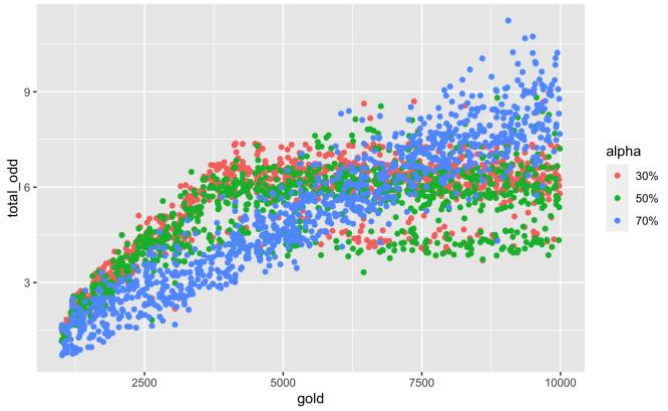


*Fig. 19 Odds Incremental versus Available In-Game Gold Grouped by Alpha Values*

It suggests the alpha value used by GRASP algorithm could be a dynamic parameter based on the in-game gold that the player has when the algorithm is generating the items recommendation. When player has less than six thousand units of in-game gold the alpha value used will be 30% and 70% otherwise.

*3) Validation Results*

The proposed rational agent architecture was also tested with the validation dataset to validate if recommendations are generated with a consistent time and odds incremental than observed during testing stage.

The Fig. 20 shows that experiments performed in validation dataset take an average of 0.387 seconds to generate a recommendation. This average is slightly higher than time in test dataset. However, it is close to real time and will not have an impact in the player experience.

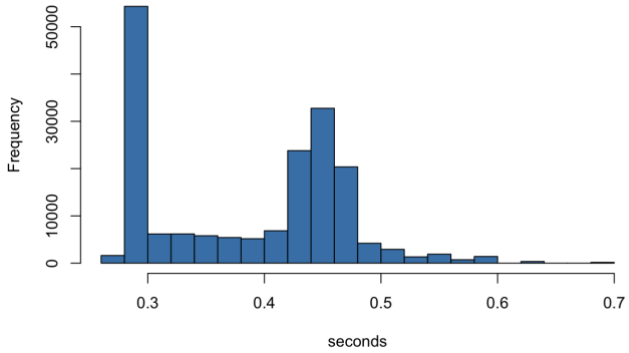


*Fig. 20 Execution Time Distribution in Validation Dataset*

Similarly, the results in the incremental odds of the experiments performed in the validation dataset are consistent with the test dataset. Fig. 21 shows the average of the incremental odds in validation dataset is 5.512.

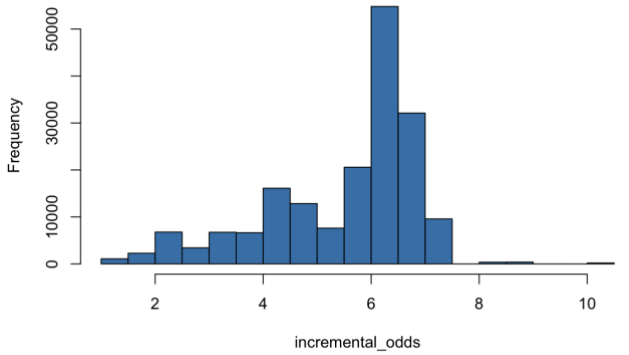


*Fig. 21 Odds Incremental Distribution in Validation Dataset*

Finally, every player into the match of the validation dataset were analyzed to evidence how many of the bought items match with the items that would be recommended by the rational agent and observe whether it has a positive impact in the winning ratio.

TABLE 4
WINNING RATIO VERSUS NUMBER OF BOUGHT ITEMS THAT MATCH WITH RECOMMENDED ITEMS

| Matching items | Number of Players | Winning Ratio |
|---|---|---|
| 0 | 28,415 | 38.1% |
| 1 | 558,843 | 44,1% |
| 2 | 1,031,233 | 64.2% |
| 3 | 193,425 | 79.4% |
| 4 | 3,133 | 68.4% |
| 5 | 2 | 0.0% |
| 6 | 109 | 48,6% |

The results show in Table 4 suggests that buying recommended items increase the odds to win a match. For instance, players with two matching items have an odds ratio of winning 3x higher than players with zero matching items. Consequently, players with three matching items have 6.5x higher odds ratio of winning than players with zero matching items.

## V. Conclusions

In this paper, a rational agent architecture was proposed. This agent can suggest to a player which item should buy during a match of a MOBA videogame to maximize his chances of winning, it considers the available resources. It was designed after exhaustive research of previous work to identify the most popular machine learning techniques, predictors used most frequently, and non-solved questions into MOBA videogames research using machine learning algorithms.

The proposed rational agent architecture has demonstrated that it could be useful for a player when making a complex decision, such as buying an item. The accuracy of prediction model would reduce him the uncertainty of the purchase, and the speed of the optimization algorithm would allow him to adapt in real-time to the state of the match given its available in-game gold.

It is possible to determine that papers using machine learning techniques to recommend items, have never used in-game gold as a constraint. It should be considered into an optimization problem; however, optimization algorithms are the less used machine learning techniques in MOBA videogames.

Also, reviewed papers were group into six categories based on their investigation objectives. It was helpful to find the variables used most frequently by category, and variables used in several categories. The transversally used variables were considered as relevant in the MOBA videogames context. Subsequently, used as predictors in prediction models. They are Items, Chosen Hero, and Roles.

Offline sensor is connected to the League of Legends API from where it was able to obtain historical information of more than five hundred thousand matches. The obtained dataset proves to be larger and more representative than several benchmark datasets.

Experiments were conducted with the four most popular machine learning prediction models: ANN, GLM, RF and SVM. The experiments showed that the most suitable prediction model is GLM. During early and mid-stages, GLM has an accuracy higher than 0.95 predicting which team will win the match.

Despite available resources have not been considered in previous investigations as a constraint, the proposed architecture considers in-game gold successfully. Recommendations of what item to buy, are generate using GRASP, an optimization algorithm never used into the MOBA videogames previous research. Several simulations were performed in this component getting a configuration which generates a recommendation in almost real-time (0.3556 second). In average, recommendations generated by GRASP increases the odds to win a match in 5.3014.

## VI. Future Work

Several opportunities to extend this research have been identified. First, to strengthen conclusions related to the components of the rational agent by testing the proposed architecture in MOBA videogames different than League of Legends. Second, to identify player's playstyle -e.g., aggressive, defensive, etc.- and use it to modify recommendations based on playstyle, it could be done adding a personalization layer into the rational agent. Third, to improve versatility of the recommendations adding different heuristics to generate recommendations, they could be added simultaneously into the agent and include a component which choose what recommendation to use. Finally, to connect individual rational agent with the teammates agent in order to generate group recommendations, it could be done with an interconnected architecture between agents.

## References

[1] W. Looi, M. Dhaliwal, R. Alhajj and J. Rokne, "Recommender system for items in dota 2," *IEEE Transactions on Games*, vol. 11, no. 4, pp. 396-404, 2018.

[2] L. Jackson, "How MOBAs Took Over Gaming," IGN Middle East, 1 August 2013. [Online]. Available: https://me.ign.com/en/pc/70142/feature/how-mobas-took-over-gaming. [Accessed 23 March 2021].

[3] Field Level Media, "Esports primer: League of Legends," Reuters, 24 March 2020. [Online]. Available: https://www.reuters.com/article/esports-lol-primer-idUSFLM1tj2jd. [Accessed 23 March 2021].

[4] J. Calixto, "Proving Grounds: The Geography of the MOBA Map," The Meta, [Online]. Available: https://killscreen.com/themeta/proving-grounds-geography-moba-map/. [Accessed 23 March 2021].

[5] A. Gies, "The Normal Person's Guide to Watching Competitive Dota 2 (2017 Edition)," Polygon, 2 August 2017. [Online]. Available: https://www.polygon.com/2017/8/2/16073588/the-normal-persons-guide-to-watching-competitive-dota-2-2017-edition. [Accessed 23 March 2021].

[6] Garamor, "Heroes of the Storm: How to Fully Utilize Mercenary Camps," Dignitas, 27 October 2017. [Online]. Available: http://team-dignitas.net/articles/blogs/Heroes-Of-The-Storm/11768/heroes-of-the-storm-how-to-fully-utilize-mercenary-camps. [Accessed 23 March 2021].

[7] K. Webb, "More than 100 million people watched the 'League of Legends' World Championship, cementing its place as the most popular esport," Insider, 18 December 2019. [Online]. Available: https://www.businessinsider.com/league-of-legends-world-championship-100-million-viewers-2019-12 [Accessed 1 May 2021].

[8] M. V. Birk, B. Buttlar, J. T. Bowey, S. Poeller, S. C. Thomson, N. Baumann and R. L. Mandryk, "The Effects of Social Exclusion on Play

Experience and Hostile Cognitions in Digital Games," in *2016 CHI Conference on Human Factors in Computing Systems*, 2016.

[9] M. Larsson, "Enhancing the Value Proposition of Live Esports Consumption with AI Technology," 2018.

[10] C. Eggert, M. Herrlich, J. Smeddinck and R. Malaka, "Classification of player roles in the team-based multi-player game dota 2," in *International Conference on Entertainment Computing*, 2015.

[11] H. Wang, H.-T. Yang and C.-T. Sun, " Thinking style and team competition game performance and enjoyment," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 7, no. 3, pp. 243-254, 2015.

[12] L. M. Costa, A. C. C. Souza and F. C. M. Souza, "An approach for team composition in league of legends using genetic algorithm," in *18th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, 2019.

[13] V. Do Nascimento Silva and L. Chaimowicz, "On the development of intelligent agents for moba games," in *14th Brazilian Symposium on Computer Games and Digital Entertainment*, 2015.

[14] H. Lie, D. Lukas, J. Liebig and R. Nayak, "A Novel Learning-to-Rank Method for Automated Camera Movement Control in E-Sports Spectating," in *Australasian Conference on Data Mining*, Singapore, 2018.

[15] M. Schubert, A. Drachen and T. Mahlmann, "Esports analytics through encounter detection," in *MIT Sloan Sports Analytics Conference*, 2016.

[16] A. Semenov, P. Romov, S. Korolev, D. Yashkov and K. Neklyudov, "Performance of machine learning algorithms in predicting game outcome from drafts in dota 2," in *International Conference on Analysis of Images, Social Networks and Texts*, 2016.

[17] M. Aung, V. Bonometti, A. Drachen, P. Cowling, A. V. Kokkinakis, C. Yoder and A. Wade, "Predicting skill learning in a large, longitudinal moba dataset," in *IEEE Conference on Computational Intelligence and Games (CIG)*, 2018.

[18] I. Porokhnenko, P. Polezhaev and A. Shukhman, "Machine learning approaches to choose heroes in dota 2," in *24th Conference of Open Innovations Association (FRUCT)*, 2019.

[19] R. Smit, "A machine learning approach for recommending items in league of legends," 2019.

[20] X. Bang, W. Huiwen and R. Zhou, "What contributes to success in MOBA games? An empirical study of Defense of the Ancients 2," *Games and Culture*, vol. 14, no. 5, pp. 498-522, 2019.

[21] W. Michael and M. Deshen, "An analysis of artificial intelligence techniques in multiplayer online battle arena game environments," in *Annual Conference of the South African Institute of Computer Scientists and Information Technologists*, 2016.

[22] R. Ani, V. Harikumar, A. K. Devan and O. S. Deepa, "Victory prediction in League of Legends using Feature Selection and Ensemble methods," in *International Conference on Intelligent Computing and Control Systems (ICCS)*, 2019.

[23] S.-K. Lee, S.-J. Hong and S.-I. Yang, "Predicting Game Outcome in Multiplayer Online Battle Arena Games," in *International Conference on Information and Communication Technology Convergence (ICTC)*, 2020.

[24] X. Lan, L. Duan, W. Chen, R. Qin, T. Nummenmaa and J. Nummenmaa, "A player behavior model for predicting win-loss outcome in MOBA games," in *nternational Conference on Advanced Data Mining and Applications*, 2018.

[25] A. Summerville, M. Cook and B. Steenhuisen, "Draft-analysis of the ancients: predicting draft picks in dota 2 using machine learning," in *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2016.

[26] A. Katona, R. Spick, V. J. Hodge, S. Demediuk, F. Block, A. Drachen and J. A. Walker, "Time to die: Death prediction in dota 2 using deep learning," in *IEEE Conference on Games (CoG)*, 2019.

[27] D. Ye, G. Chen, P. Zhao, F. Qiu, B. Yuan, W. Zhang, S. Chen, M. Sun, X. Li, S. Li, J. Liang, Z. Lian, B. Shi, L. Wang, T. Shi, Q. Fu, W. Yang and L. Huang, "Supervised Learning Achieves Human-Level Performance in MOBA Games: A Case Study of Honor of Kings," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1-11, 2020.

[28] I. Makarov, D. Savostyanov, B. Litvyakov and D. I. Ignatov, "Predicting winning team and probabilistic ratings in “Dota 2” and “Counter-Strike: Global Offensive” video games.," in *International Conference on Analysis of Images, Social Networks and Texts*, Cham, 2017.

[29] M. Anshori, F. Mar'i, M. W. Alauddin and F. A. Bachtiar, "Prediction result of dota 2 games using improved svm classifier based on particle swarm optimization," in *International Conference on Sustainable Information Engineering and Technology (SIET)*, 2018.

[30] D.-H. Kim, C. Lee and K.-S. Chung, "A confidence-calibrated moba game winner predictor," in *IEEE Conference on Games (CoG)*, 2020.

[31] Z. Cleghern, S. Lahiri, O. Özaltin and D. L. Roberts, "Predicting future states in dota 2 using value-split models of time series attribute data," in *12th International Conference on the Foundations of Digital Games*, 2017.

[32] P. Goyal, A. Sapienza and E. Ferrara, "Recommending teammates with deep neural networks.," in *29th on Hypertext and Social Media*, 2018.

[33] S.-J. Hong, S.-K. Lee and S.-I. Yang, "Champion Recommendation System of League of Legends," in *International Conference on Information and Communication Technology Convergence (ICTC)*, 2020.

[34] Z. Chen, T.-H. D. Nguyen, Y. Xu, C. Amato, S. Cooper, Y. Sun and M. S. El-Nasr, "The art of drafting: a team-oriented hero recommendation system for multiplayer online battle arena games," in *12th ACM Conference on Recommender Systems*, 2018.

[35] M. Z. A. Z. Aznin, N. M. Diah and N. A. S. Abdullah, "Expert system for dota 2 character selection using rule-based technique," in *International Visual Informatics Conference*, 2019.

[36] T. D. Do, D. S. Yu, S. Anwer and S. I. Wang, "Using Collaborative Filtering to Recommend Champions in League of Legends," in *IEEE Conference on Games (CoG)*, 2020.

[37] A. Villa, V. Araujo, F. Cattan and D. Parra, "Interpretable Contextual Team-aware Item Recommendation: Application in Multiplayer Online Battle Arena Games," in *Fourteenth ACM Conference on Recommender Systems*, 2020.

[38] L. Zhang, C. Xu, Y. Gao, Y. Han, X. Du and Z. Tian, "Improved Dota2 lineup recommendation model based on a bidirectional LSTM," *Tsinghua Science and Technology*, vol. 25, no. 6, pp. 712-720, 2020.

[39] S. Russell and P. Norvig, Artificial intelligence: a modern approach, 2002.

[40] M. P. Silva, V. Do Nascimento Silva and L. Chaimowicz, "Dynamic difficulty adjustment on MOBA games," *Entertainment Computing*, vol. 18, pp. 103-123, 2017.

[41] N. P. H. Pratama, S. M. S. Nugroho and E. M. Yuniarno, "Fuzzy controller based AI for dynamic difficulty adjustment for defense of the Ancient 2 (DotA2)," in *International Seminar on Intelligent Technology and Its Applications (ISITIA)*, 2016.

[42] V. Do Nascimento Silva and L. Chaimowicz, "A tutor agent for moba games," in *XIV Brazilian Symposium on Computer Games and Digital Entertainment*, 2015.

[43] RIOT Games, "About the RIOT Games API," RIOT Games, [Online]. Available: https://developer.riotgames.com/. [Accessed 1 October 2021].

[44] RIOT Games, "Champion JSON," [Online]. Available: http://ddragon.leagueoflegends.com/cdn/11.19.1/data/en_US/champion.json. [Accessed 3 October 2021].

[45] RIOT Games, "Items Endpoint," [Online]. Available: http://ddragon.leagueoflegends.com/cdn/11.19.1/data/en_US/item.json. [Accessed 3 October 2021].

[46] M. J, "(LoL) League of Legends Ranked Games," Kaggle, [Online]. Available: https://www.kaggle.com/datasnaek/league-of-legends. [Accessed 3 October 2021].

[47] C. Ephron, "League of Legends Competitive matches, 2015 to 2018," Kaggle, [Online]. Available: https://www.kaggle.com/chuckephron/leagueoflegends. [Accessed 3 October 2021].

[48] m. fanboi, Kaggle, [Online]. Available: https://www.kaggle.com/bobbyscience/league-of-legends-diamond-ranked-games-10-min. [Accessed 3 October 2021].

[49] P. Campanelli, "League of Legends Ranked Matches," Kaggle, [Online]. Available: https://www.kaggle.com/paololol/league-of-legends-ranked-matches. [Accessed 3 October 2021].

[50] J. G. López Guzmán and C. J. Bustacara Medina, "Relevant Independent Variables on MOBA Video Games to Train Machine Learning Algorithms," in *International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 2021*, Plzen, 2021.

[51] A. Drachen, M. Yancey, J. Maguire, D. Chu, I. Y. Wang, T. Mahlmann, M. Schubert and D. Klabajan, "Skill-based differences in spatio-temporal team behaviour in defence of the ancients 2 (dota 2)," in *IEEE Games Media Entertainment*, 2014.

[52] F. Block, V. Hodge, S. Hobson, N. Sephton, S. Devlin, M. F. Ursu, A. Drachen and P. I. Cowling, "Narrative bytes: Data-driven content production in esports.," in *ACM international conference on interactive experiences for TV and online video*, 2018.

[53] A. P. Afonso, M. B. Carmo and T. Moucho, "Comparison of Visualization Tools for Matches Analysis of a MOBA Game," in *23rd International Conference Information Visualisation (IV)*, 2019.

[54] L. Yu, D. Zhang, X. Chen and X. Xie, "Moba-slice: A time slice based evaluation framework of relative advantage between teams in moba games," in *Workshop on Computer Games*, 2018.

[55] Q. Li, P. Xu, Y. Y. Chan, Y. Wang, Z. Wang, H. Qu and X. Ma, "A visual analytics approach for understanding reasons behind snowballing and comeback in moba games," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 211-220, 2017.

[56] A. Sapienza, H. Peng and E. Ferrara, "Performance dynamics and success in online games," in *IEEE International Conference on Data Mining Workshops (ICDMW)*.

[57] S. Demediuk, A. Murrin, D. Bulger, M. Hitchens, A. Drachen, W. L. Raffe and M. Tamassia, "Player retention in league of legends: a study using survival analysis," in *Australasian Computer Science Week Multiconference*, 2018.

[58] O. Cavadenti, V. Codocedo, J.-F. Boulicaut and M. Kaytoue, "What did i do wrong in my MOBA game? Mining patterns discriminating deviant behaviours," in *IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 2016.

[59] M. Hirth, K. Borchert, F. Allendorf, F. Metzger and T. Hoßfeld, "Crowd-based Study of Gameplay Impairments and Player Performance in DOTA 2," in *4th Internet-QoE Workshop on QoE-based Analysis and Management of Data Communication Networks*, 2019.

[60] J. Eichelbaum, H. Roony and H. Olaf, "Classification of Icon Type and Cooldown State in Video Game Replays," in *International Conference Image Analysis and Recognition*, 2018.

[61] L. Wang, Y. Tang and J. Liu, "WPQA: A Gaming Support System Based on Machine Learning and Knowledge Graph," in *Joint International Semantic Technology Conference*, Singapore, 2019.

[62] T. Tsogbadrakh and M. Spradling, "Genetic Approach to Stable Partitioning in Online Role Based Hedonic Games," in *IEEE Congress on Evolutionary Computation (CEC)*, 2019.

[63] T. Gonçalves, P. Vieria, A. P. Alfonso, M. B. Carmo and T. Moucho, "Analysing player performance with animated maps," in *22nd international conference information visualisation (IV)*, 2018.

[64] S. Ahmad, A. Bryant, E. Kleinman, Z. Teng, T.-H. D. Nguyen and M. S. El-Nasr, "Modeling Individual and Team Behavior through Spatio-temporal Analysis," in *Annual Symposium on Computer-Human Interaction in Play*, 2019.

[65] S. Kim, D. Kim, H. G. Ahn and B. Ahn, "Implementation of user playstyle coaching using video processing and statistical methods in league of legends," *Multimedia Tools and Applications*, pp. 1-13, 2020.

[66] E. Kleinman, S. Ahmad, Z. Teng, A. Bryant, T.-H. D. Nguyen, C. Harteveld and M. S. El-Nasr, ""'And then they died": Using Action Sequences for Data Driven, Context Aware Gameplay Analysis," in *International Conference on the Foundations of Digital Games*, 2020.

[67] D. Gordeau and L. Archambault, "Discriminative neural network for hero selection in professional Heroes of the Storm and DOTA 2," *IEEE Transactions on Games*, 2020.

[68] C. Ringer, J. A. Walker and M. Nicolaou, "Multimodal joint emotion and game context recognition in league of legends livestreams," in *IEEE Conference on Games (CoG)*, 2019.

[69] L. Hanke and L. Chaimowicz, "A recommender system for hero line-ups in MOBA games," in *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2017.

[70] M. Mora-Cantallops and M.-Á. Sicilia, "Player-centric networks in League of Legends," *Social Networks*, vol. 55, pp. 149-159, 2018.

[71] F. F. do Nascimento, A. S. da Costa Melo, I. B. da Costa and L. B. Marinho, "Profiling successful team behaviors in League of Legends," in *23rd Brazillian Symposium on Multimedia and the Web*, 2017.

[72] J. G. Reitman, "Distributed cognition and temporal knowledge in league of legends," *International Journal of Gaming and Computer-Mediated Simulations (IJGCMS)*, vol. 10, no. 1, pp. 23-41, 2018.

[73] Z. Chen, Y. Yang, C. Tan, D. Cheng, A. Chen and Y. Zhuang, "What makes a good team? A large-scale study on the effect of team composition in Honor of Kings," in *The World Wide Web Conference*, 2019.

[74] N. Wang, L. Li, L. Xiao, G. Yang and Y. Zhou, "Outcome prediction of dota2 using machine learning methods," in *International Conference on Mathematics and Artificial Intelligence*, 2018.

[75] M. Mora-Cantallops and M.-Á. Sicilia, "Team efficiency and network structure: The case of professional League of Legends," *Social Networks*, vol. 58, pp. 105-115, 2019.

[76] V. Araujo, F. Rios and D. Parra, "Data mining for item recommendation in MOBA games," in *13th ACM Conference on Recommender Systems*, 2019.

[77] B. Xia, H. Wang and R. Zhou, "What contributes to success in MOBA games? An empirical study of Defense of the Ancients 2," *Games and Culture*, vol. 14, no. 5, pp. 498-522, 2019.

[78] A. R. Novak, K. J. Bennett, M. A. Pluss and J. Fransen, "Performance analysis in esports: modelling performance at the 2018 League of Legends World Championship," *International Journal of Sports Science & Coaching*, vol. 15, no. 5-6, pp. 809-817, 2020.

[79] A. Kica, A. La Manna, L. O'Donnell, T. Paolillo and M. Claypool, "Nerfs, Buffs and Bugs - Analysis of the Impact of Patching on League of Legends," in *2016 International Conference on Collaboration Technologies and Systems (CTS)*, Orlando, FL, USA, 2016.

[80] L. Hwang and K. Yoon, Multiple Attribute Decision Methods and Applications, Berlin: Springer, 1982.

[81] R. M. Karp, "Reducibility Among Combinatorial Problems," *Complexity of computer computations*, pp. 85-103, 1972.

[82] T. A. Feo and M. G. Resende, "A probabilistic heuristic for a computationally difficult set covering problem," *Operations Research Letters*, vol. 8, no. 2, pp. 67-71, 1989.

[83] Databricks, "Databricks Runtime 8.2 for Machine Learning," Databricks, 22 April 2021. [Online]. Available: https://docs.databricks.com/release-notes/runtime/8.2ml.html. [Accessed 16 May 2021].

[84] S. A. Halim, Y. Indrianti, Y. Udjaja, J. Moniaga and B. A. Makalew, "The Repercussions of Game Multiplayer Online Battle Arena," in *International Conference of Artificial Intelligence and Information Technology (ICAIIT)*, 2019.

[85] M. Waltham and D. Moodley, "An analysis of artificial intelligence techniques in multiplayer online battle arena game environments," in *Annual Conference of the South African Institute of Computer Scientists and Information Technologists*, 2016.