Continuous Outcome Prediction of League of Legends Competitive Matches Using Recurrent Neural Networks

Antonio Luis Cardoso Silva, Gisele Lobo Pappa and Luiz Chaimowicz
Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
Belo Horizonte, MG, Brazil
{*antoniosilva, glpappa, chaimo*}*@dcc.ufmg.br*

***Abstract*—League of Legends (LoL) is currently the most popular video game, reaching 7.5 millions of players playing simultaneously at peak hours. LoL has a competitive scene with regional and international tournaments, offering prizes of more than 5 million dollars. In this paper, we compare different recurrent neural networks in the task of predicting the result of a competitive match using as input information of what happened in a specific minute into the game. This kind of network can be used to identify the power spikes of different team compositions. While other works focus on predicting the result given information only from the characters picked to the match, there is no previous work attempting this kind of prediction in MOBA games. For this task, we discover that a simple RNN is the more effective recurrent neural network and we obtain an accuracy of 63.91% when using data from minutes 0 to 5 and 83.54% when using data from minutes 20 to 25.**

***Keywords*-League of Legends, Recurrent Neural Networks, Prediction.**

I. Introduction

League of Legends (LoL) is a multiplayer online battle arena (MOBA) video game developed by Riot Games in 2009. In its common gameplay, five players fight another group controlling characters called champions in a map with three lanes and a jungle. There are many objectives that should be destroyed during the game such as towers, barons, dragons, that gives buffs to champions and allow the main objective, called nexus, to be destroyed.

In LoL matches, specially competitive ones, each team has win conditions that depend on the champions picked, gold advantage, or destruction of objectives. An example of win condition is when a team kills an objective called Baron which makes the champions stronger for a certain period of time. Another is when a champion that is good late in the game buys several items, so the team starts grouping and fighting taking advantage of their power spike.

In this work, we use information on what happened in a specific minute of a match and try to predict the outcome of a game using different types of Recurrent Neural Networks (RNNs). We hope that this kind of neural network can be used by teams to identify possible win conditions according to champions being played, gold advantage, number of towers destroyed, among other features. To our knowledge, there is no previous research that uses in-game data to predict the outcome of LoL or other MOBA games. In general, previous research focuses on predicting the result using information available before the game starts [1].

The dataset used in this work is available on *Kaggle* and contains information about 7621 regional and international competitive matches of League of Legends. It contains data from every minute of a match, including: gold advantage, towers destroyed, champions killed, besides the general information of the game, such as, teams in the match, champions chosen, name of the player, etc. The data is formatted in a way that each input represents data from a single minute of a game and the output is the result of the game. Due to the sequential nature of the data, we use Recurrent Neural Networks to predict the winner of a match using data between minutes 0 and 25. As will be presented, using a simple RNN, we obtained an accuracy between 63.91% and 83.56% depending on how late in the match the data came from, which is expected since the winning team will accumulate more advantage later in the match.

This paper is organized as follows: Section II gives a background of the game League of Legends and some of the techniques used in this paper and also discusses related works; Section III describes the methodology used in this work; Section IV shows the results of the experiments and Section V summarizes our conclusions and discusses possible future works.

II. Background and Related Work

A. League of Legends

League of Legends is a Multiplayer Online Battle Arena (MOBA) game developed by Riot Games released in 2009. MOBAs are characterized by their online competitive nature, where two teams of players compete against each other aiming to destroy a structure called nexus in the enemy base. A typical map of a MOBA game can be seen in Fig. 1. There are three lanes: top, middle and bottom represented in yellow, where minions from each team walk to confront each other. There is also a jungle, represented in green, containing monsters that can be killed for gold and buffs. The map also

contains towers, represented in blue and red dots, which defend an area from enemy attacks. The most played mode in LoL is the one in which the teams are composed of five players, each one controlling a different character called champion or hero. They are chosen in the beginning of a match from a champion pool.

Each champion has unique skills that can be used against other champions, minions or monsters. Minions spawn in waves regularly from the base of each team and walk in all the three lanes until finding minions from the other team. This is when the battle starts. Champions acquire gold killing enemies and the gold can be used to buy new items. Items improve the attributes of the champions making them do more damage or increasing their defense.

The main objective in LoL is to destroy a structure inside each team's base called nexus, but to be able to do that, all towers and the inhibitor in one lane have to be destroyed beforehand. Towers are structures that inflict damage to enemies inside their radius. A tower prioritizes attacking minions and enemy champions, if an allied champion is attacked inside its range. Inhibitors are structures that, when destroyed by the enemy team, make enemy minions more powerful and allow the nexus to be destroyed.

When choosing a champion, players have to be aware of the role they are going to play in the team. Different champions may have one or more roles, but most of them performs better in the main role they were designed. The player also has to be aware of the current metagame. The metagame dictates the types of champions that are more adequate in each position in the map and also which champions are the strongest in the game. Due to the constant updates of the game, performed through patches, the metagame tends to change in every balance change. An example of metagame is the most common team compilation in LoL: normally two champions go to bottom lane, one Support and one Attack Damage Carry (ADC), in the middle lane there will be a Mage or Assassin, in the Jungle/Ganker role goes a Tank or a Fighter and in the top lane the champion can be a Initiator, Front-liner or Tank.

B. Recurrent Neural Networks

In a standard model of artificial neural networks, data flows from the input layer, through the hidden layers to the output layer, so it never passes through a node twice. Because of that, the network has no memory of the input they received previously and has no notion of the input order. In Recurrent Neural Networks (RNN), the information cycles go through a loop, taking into consideration the current input and also what it learned from previous inputs. This configuration of neural networks allows the presence of a short-term memory and is more suitable for tasks that involve sequential data, such as time series, text, financial data, audio, among others.

One specific type of RNN is the Long Short-Term Memory (LSTM) network which allows a RNN to extend its memory making them more suitable for tasks where there is a lag between important experiences. LSTMs have special units similar to gated cells that decide which information to store or to forget according to its importance [2]. The importance is determined through weights that are also learned by backpropagation. This means that the network learns which information is more important than others during training. The gated cells can be of one of three different types: input, forget, and output gates. The input gate defines what information from the input can pass through, the forget gate deletes information that is not important and the output gate defines how much impact the output has considering the input at the current time step.

Another type of RNN is the Gated Recurrent Unit (GRU), which is similar to the LSTM but has only two types of gates: update and reset gates [3]. The update gate determines how much information from the past needs to pass to the future and the reset gate determines how much of the information has to be forgotten.

C. Related Work

To the best of our knowledge there is no previous work that tries to predict results in MOBA games with neural networks using information collected during the match. Researchers mostly focus on predicting results using information available prior to the beginning of a game. For example, in [4], the authors use information of heroes picked and their win rates in DOTA 2 matches and predict the outcome using Logistic Regression and Random Forests with 73% accuracy. Similar algorithms are used to predict the outcome in League of Legends in different time intervals, obtaining in average 75% of accuracy [5]. Another work also tries to predict the outcome of DOTA 2 matches according to the heroes picked, but this time using the Naive Bayes classifier and obtaining an accuracy of 58.99% [1]. In Hanke et al. [6], a hero recommendation system using association rules was developed for DOTA 2 and a neural network was created to predict the result of matches according to the heroes picked. The neural network obtained an accuracy of 88.63% in the test set. Neural networks were also used for predicting results in other sports. For example, in [7], authors use information of box scores in NBA matches to predict the winning team with an accuracy of 74.33%.

III. Methodology

Our methodology consists in training a RNN with data from a LOL match dataset and evaluating its performance in predicting the match outcome. We first perform some preliminary experiments to define the network architecture. Once this is done, three different types of recurrent networks are implemented and trained: a simple RNN, a LSTM and a GRU. The network with the best performance is trained again using different time intervals so we can investigate

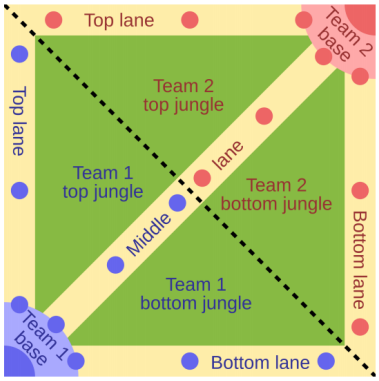

Figure 1. A typical map of a MOBA game.

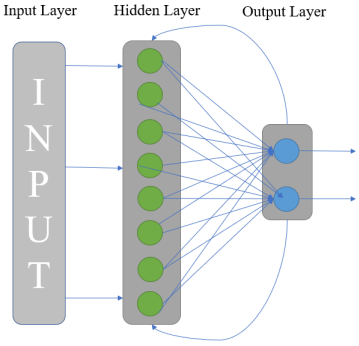

Figure 2. Topology of the network used.

how the data from different periods of a match impacts the prediction accuracy.

The dataset used in this work is available on Kaggle[1] and contains information from 7621 competitive LoL matches played between 2015 and early 2018. It includes matches from several tournaments such as: NALCS (North America), LCK (Korea), CBLoL (Brazil), among others. The dataset has 53 dimensions which include general information from the match such as:

1) The tournament the match is happening;
2) Teams involved;
3) Name of the players on each team;
4) Champions chosen;
5) Result of the match.

It has also information about what happened on each minute in a match, such as:

1) Amount of gold earned by each champion;
2) Minute a champion was killed;
3) Minute a objective was destroyed.

Due to the sequential nature of the data, a Recurrent Neural Network was chosen for the task of predicting the outcome of a match given what happened in a specific minute of the game. In the dataset, all information from the match was condensed in a single instance. To be able to use the dataset in a RNN, the data was converted so each instance represented information from a minute in the match. To do that, for each match we created N new instances where N is the number of minutes of the match. General information from the match was kept the same in all N instances but specific information for a minute i in the game was put in the ith instance. New dimensions were created to represent the number of kills and deaths each champion had in the ith minute and also to represent the gold difference between each position. After the preprocessing is done, the result was a new dataset with 280603 instances and 65 dimensions. The RNNs were implemented in Python using Keras and Theano libraries.

[1]https://www.kaggle.com/chuckephron/leagueoflegends

IV. Experiments and Results

Initially, a Long Short Term Memory (LSTM) Network was chosen for the task. So there was a initial experiment to discover which network topology would be more suitable for the problem. First, we tested complex networks with two hidden layers with 64 units, however there was overfitting obtaining poor accuracy so the complexity was decreased until the accuracy was satisfactory. The final topology is described below and shown in Fig. 2. The same topology was used for all types of RNN in the next experiments.

- One hidden layer with 8 recurrent units
  - Regularizer: Dropout 0.25
- One output layer with 2 units
  - Activation: Softmax
- Time steps: 9
- Loss function: Binary Cross-Entropy
- Optimizer: RMSProp

After these preliminary experiments, we implemented different types of recurrent networks and evaluated their accuracy. As mentioned, we implemented three types of RNN: LSTM, Gated Recurrent Unit (GRU), and a simple RNN using a batch size of 64 and 50 epochs. For the experiments, 67% of the matches were chosen for training and 33% for test. From those 33%, only the instances from the minutes 10 and 15 were chosen. Each test was executed five times, and the average and standard deviation are shown in table I. We can observe that SimpleRNN obtained higher accuracy. This is probably because the problem is not very difficult, and a simple RNN is sufficient for the prediction. On the other hand, more complex networks such as LSTM and GRU may require more data to have better performance. Fig. 3 shows the error in training and test for the SimpleRNN which shows that the topology for this network type was not causing overfitting, since the test and training errors kept decreasing. When there is overfitting, we expected to see the test error increasing while the training error decreases.

Table I
RESULTS COMPARING DIFFERENT RNNS FOR THE TIME INTERVAL 10-15

Network	SimpleRNN	GRU	LSTM
Accuracy	76.29%	72.92%	71.63%
Standard Deviation	1.16%	1.39%	1.63%

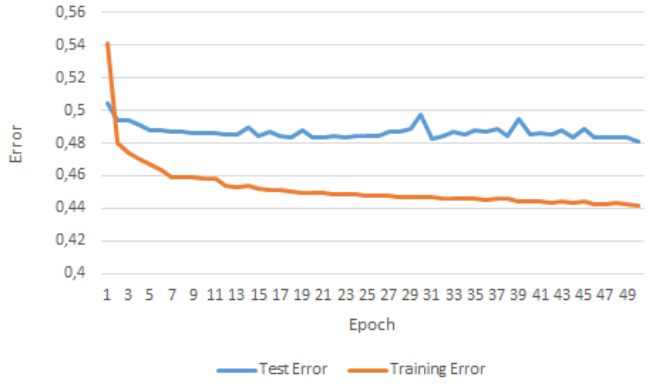

Figure 3. Comparison between error in training and testing for the SimpleRNN

Table II
RESULTS OBTAINED FOR DIFFERENT TIME INTERVALS USING SIMPLERNN

Minutes	0-5	5-10	10-15	15-20	20-25
Accuracy	63.91%	68.69%	75.23%	80.18%	83.54%
Std. Dev.	1.81%	1.31%	1.02%	1.05%	1.28%

Since it had the best performance, the SimpleRNN was chosen for the next set of experiments, which evaluate the impact of training a RNN with data from different game intervals. We used five different time intervals: from 0 to 5 minutes, 5 to 10, 10 to 15, 15 to 20, 20 to 25. It is worth noting that the average match time in the database is 37 minutes. Training was made using k-fold cross validation with k equal to 5. Although in the dataset each instance represents a minute in a match, the split between training and test set is made considering whole matches so testing is done with matches that the network have not seen. The result is shown in table II. The accuracy increasing when information from later in the game is passed to the network is expected because the winning team gain advantage in gold, kills, towers and this difference is more clear the long the game goes. The may challenge encountered to reach better accuracy is the change of meta-game over time with the game receiving constant updates: new champions being added and existing ones being tweaked or completely reworked.

V. CONCLUSION

In this work different types of RNNs were compared in the task of predicting the outcome of competitive matches in League of Legends using as input information on what happened in specific time intervals of the match. A Simple RNN obtained higher accuracy and was chosen for new experiments with different time intervals. The RNN obtained results varying from 63.91% and 83.54% of accuracy depending on the time interval of the game. A possible use for this kind of network is to analyze power spikes of teams composition and identify when they should fight or just accumulate gold and experience until they are strong enough. For future work, we may increase the dataset by using data from ranked matches in the Challenger ladder where the best players play against each other and is a close representation of competitive matches. It is possible that there is irrelevant dimensions being used, so we can work to eliminate them. We also want to investigate the use of other types of networks and Machine Learning algorithms to predict results of matches.

ACKNOWLEDGMENTS

This work was partially supported by Fapemig, CAPES, and CNPq.

REFERENCES

[1] K. Wang and W. Shang, "Outcome prediction of DOTA2 based on naïve bayes classifier," in *16th IEEE/ACIS International Conference on Computer and Information Science, ICIS 2017, Wuhan, China, May 24-26, 2017*, 2017, pp. 591–593. [Online]. Available: https://doi.org/10.1109/ICIS.2017.7960061

[2] F. A. Gers, J. Schmidhuber, and F. A. Cummins, "Learning to forget: Continual prediction with LSTM," *Neural Computation*, vol. 12, no. 10, pp. 2451–2471, 2000. [Online]. Available: https://doi.org/10.1162/089976600300015015

[3] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," in *Proceedings of SSST@EMNLP 2014, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, Doha, Qatar, 25 October 2014*, 2014, pp. 103–111. [Online]. Available: http://aclweb.org/anthology/W/W14/W14-4012.pdf

[4] N. Kinkade, L. Jolla, and K. Lim, "Dota 2 win prediction," Technical report, University of California, San Diego, Tech. Rep., 2015.

[5] R. T. Souza, "Aplicação de algoritmos classificadores para previsão de vitória em uma partida de League of Legends," Rio Grande do Sul, p. 21, 2017.

[6] L. Hanke and L. Chaimowicz, "A recommender system for hero line-ups in MOBA games," in *Proceedings of the Thirteenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE-17)*, 2017, pp. 43–49. [Online]. Available: https://aaai.org/ocs/index.php/AIIDE/AIIDE17/paper/view/15902

[7] B. Loeffelholz, E. Bednar, and K. W. Bauer, "Predicting nba games using neural networks," *Journal of Quantitative Analysis in Sports*, vol. 5, no. 1, 2009.