League of Legends Match Outcome Prediction using Deep Learning Techniques

James Ting

May 22, 2020

Introduction

League of Legends is a popular and competitive Multiplayer Online Battle Arena, where 2 teams of 5 players compete to destroy the enemy Nexus (Figure 1a & Figure 1b). Each player controls a champion (a character), and to win, the team must destroy a series of enemy turrets, invade the enemy base, and destroy the opponents Nexus.

As the players play each match, their champion will level up and earn gold, which the player can then exchange at the shop for items that can strengthen each character. Players also earn gold from: kills, killing sprees, enemy minions, and neutral monsters.

Throughout the map, there are buffs that each team can earn by killing neutral monsters. These neutral monsters also provide gold to the player who earns the kill, which allows the players to obtain more items.

While Riot Games (the company behind the game) does try to ensure that the win rate for each team is as close to 50% as possible, it is well known that the blue team actually wins approximately 55% of games. Riot Games compensates this by ensuring red team players are slightly higher skilled, but this method is not foolproof, since as will be demonstrated later, the network is able to predict which team will win with reasonable accuracy for the given information.

League of Legends has, at the time of writing, approximately 120 million monthly players. Furthermore, Riot Games has an API that anyone can use in order to pull data about any player, and their match history. As such, there is a massive wealth of data that is available and makes it perfect for a machine learning project. **In this project, I will use a binary classification neural network to predict match outcomes on two types of data: pre-match data and post-match data**. Pre-match data will be data that is available prior to the start of the match, and post-match data will also include each players performance data for this specific match.

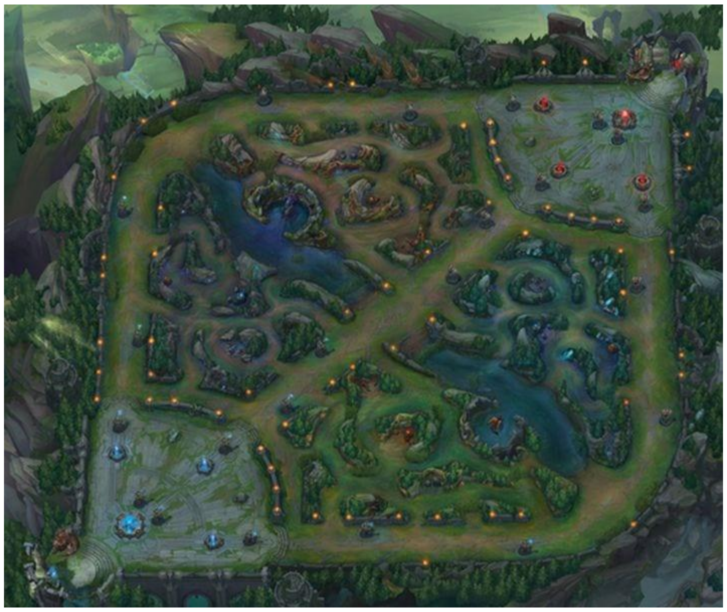

Figure 1a. Summoners Rift Map. The 2 Nexus can be seen in the top-right and bottom left corners of the map (Summoners Rift, 2020)

Figure 1b.: The blue team Nexus (Nexus, 2020)

Data Collection

The data was collected using the RiotWatcher Python Library (RiotWatcher, 2019). I built classes to help facilitate the collection of data. The first class, the AbstractDataPuller (and its sub classes) will be the objects that will make calls to the RiotWatcher library to get the information from the API. It contains three subclasses: PlayerDataPuller, MatchDataPuller, ChampionMasteryDataPuller. Each subclass is responsible for pulling data for players, matches, and champion mastery respectively.

The next class is the MatchCrawler class. It is essentially an iterator, that has 2 functions: hasNext() and next() (similar to a Java Iterator Object). hasNext() simply returns a boolean representing whether the specified number of data points has been added to the data set. next() returns the next matchID to pull the data for. next() selects the next match ID by taking the current match ID, and for a random summoner in that match, gets a non-empty match list. If that match list is empty, then it moves to the next summoner. Once a non-empty match list has been found, it will return a random matchID in the list.

The final class is the DataSetMaker class. This class makes use of the two classes, and writes each data points to the file appropriately. It is also a object class, which will be created by the main method in the Driver.py file.

For one reason or the other, the DataSetMaker writes match stats fastest when writing 3 lines at a time. Therefore, in Driver.py, the user can write how many batches of three they wish to input, and then the program will run accordingly.

Further, Driver.py includes an option to verify the data. This is done by checking that each matchID in the data file is unique.

The Riot API does set limitations on its users. Without an application and a development API key, users are limited to 20 requests every 1 second, and 100 requests every 2 minutes. Further, sometimes the Riot API will return null data for a summoner who appeared in a game, making it impossible to retrieve that summoners match data. As a result, the collection of data is severely limited. However, at the time of writing, approximately 5 454 matches have been collected. The goal is to collect at least 10 000 data points, to get a larger data set than was obtained by Kenneth Hall, who conducted a similar project with a data set of 1000 matches.

Match Data

For each match, a total of 775 different features was collected.

For each team, the following data was collected:

- **First Blood**: Boolean representing whether the team got first kill in a game
- **First Tower**: Boolean representing whether the team destroyed the first tower in a game
- **First Inhibitor**: Boolean representing whether the team destroyed the first inhibitor (a structure in the enemy base that must be destroyed to reach the Nexus) in a game
- **First Baron**: Boolean representing whether the team obtained the first Baron, a neutral monster that gives the entire team buffs

- **First Dragon**: Boolean representing whether the team obtained the first Dragon, a neutral monster that gives the entire team buffs
- **First Rift Herald**: Boolean representing whether the team obtained the first Rift Herald, a neutral monster that allows the team to spawn a tower destroying monster
- **Tower Kills:** Integer representing the number of tower kills
- **Inhibitor Kills:** Integer representing the number of inhibitor kills
- **Baron Kills:** Integer representing the number of Baron kills
- **Dragon Kills:** Integer representing the number of Dragon kills
- **Rift Herald Kills:** Integer representing the number of Rift Herald kills

For each summoner, a total of 75 different stats were collected. The full list can be found in Appendix 1a. The stats range from number of kills obtained, to the amount of gold, to the items that each player bought.

Finally, 2 items were appended to the end of the row. If it was a blue team victory, then a 1 and a 0 was appended to the end, and a 0 and a 1 was appended otherwise.

Data Preparation

The data was prepared by removing unnecessary columns through the use of the ExcludeColumnsGenerator class. This file contained a class which represents an object that will, given the full set of columns and the set of columns to be excluded, will give the indices of columns that are to be excluded. This can then be used by the DataReaderClass (located in the Model Jupyter Notebook) to load the data from the file and then pass it to the network for training.

There are two type of models that can be trained. Models that predict the outcome before the match occurs, which use pre-match data, and models that predict the outcome after the match occurs, which use post-match data. The ExcludeColumnsGenerator can return the indexes of both types of data for training.

The following data is never used for training:

- accountId
- role
- lane

Furthermore, the team stats are only included in the post-match data.

The columns that are used for pre-match and post-match data are included in Appendix 1b and 1c respectively.

The dataset is split into the training set and the validation set. The validation set is 10% of the entire data set, and the other 90% is the training set.

The Network

The network was build using the PyTorch library. It has 1 hidden layer, and has a dropout rate of 10%. It used a RELu as a activation function, and the Adam optimizer with a learning rate of 0.001. It was based on the network with the best performance that was found by Kenneth Hall's project. The model was trained with layer sizes of 32, 64, 128, 256, 512, 1024 and, 2048. It was found that the network with a hidden layer of 128 and 256 have the best performance and have the smoothest decrease in validation accuracy.

Results

Pre-match data

When using the pre-match data, the network can obtain approximately 65% validation accuracy, which while better than guessing blue team will win every time, isn't good enough to place bets on. It is worth noting however, that the data set collect represents only a small fraction of the number of games played each day. While it is possible for this to simply be the result of a data set being to small to capture the entire relationship, Kenneth Hall and this project demonstrate that there is more work to be done in ensuring the win rate of each team is fair and balanced

When using the post-match data, the network can obtain a validation accuracy of approximately 95%. This should come as no surprise however, as the information about how many kills each player got, how many buffs, how many towers destroyed, and all other stats are indicative of which team will win.

Conclusion

Appendix

Appendix 1a: List of all player stats collected for each match. Each summoner stat is written as it appears in the

- accountId
- summoner level
- role
- lane
- championLevel
- championPoints
- lastPlayTime
- championPointsSinceLastLevel
- championPointsUntilNextLevel
- chestGrantedtokensEarned
- totalChampionMastery
- championId
- spell1Id
- spell2Id
- item0
- item1
- item2
- item3
- item4
- item5
- item6
- kills
- deaths
- assists
- largestKillingSpree
- largestMultiKill
- killingSprees
- longestTimeSpentLiving
- doubleKills
- tripleKills
- quadraKills
- pentaKills
- totalDamageDealt
- magicDamageDealt
- physicalDamageDealt
- trueDamageDealt
- largestCriticalStrike
- totalDamageDealtToChampions
- magicDamageDealtToChampions

- physicalDamageDealtToChampions
- trueDamageDealtToChampions
- totalHeal
- totalUnitsHealed
- damageSelfMitigated
- damageDealtToObjectives
- damageDealtToTurrets
- visionScore
- timeCCingOthers
- totalDamageTaken
- magicalDamageTaken
- physicalDamageTaken
- trueDamageTaken
- goldEarned
- goldSpent
- turretKills
- inhibitorKills
- totalMinionsKilled
- neutralMinionsKilled
- neutralMinionsKilledTeamJungle
- neutralMinionsKilledEnemyJungle
- totalTimeCrowdControlDealt
- champLevelvisionWardsBoughtInGame
- sightWardsBoughtInGamewardsPlaced
- wardsKilled
- firstBloodKill
- firstBloodAssist
- firstTowerKill
- firstTowerAssist
- combatPlayerScore
- objectivePlayerScore
- totalPlayerScore
- totalScoreRank

Appendix 1b: List of pre-match data for each summoner

- summoner level
- championLevel
- championPoints
- lastPlayTime
- championPointsSinceLastLevel
- championPointsUntilNextLevel
- chestGrantedtokensEarned
- totalChampionMastery

- championId

Appendix 1c: List of post-match data for each summoner

- summoner level
- championLevel
- championPoints
- lastPlayTime
- championPointsSinceLastLevel
- championPointsUntilNextLevel
- chestGrantedtokensEarned
- totalChampionMastery
- championId
- spell1Id
- spell2Id
- item0
- item1
- item2
- item3
- item4
- item5
- item6
- kills
- deaths
- assists
- largestKillingSpree
- largestMultiKill
- killingSprees
- longestTimeSpentLiving
- doubleKills
- tripleKills
- quadraKills
- pentaKills
- totalDamageDealt
- magicDamageDealt
- physicalDamageDealt
- trueDamageDealt
- largestCriticalStrike
- totalDamageDealtToChampions
- magicDamageDealtToChampions
- physicalDamageDealtToChampions
- trueDamageDealtToChampions
- totalHeal
- totalUnitsHealed

- damageSelfMitigated
- damageDealtToObjectives
- damageDealtToTurrets
- visionScore
- timeCCingOthers
- totalDamageTaken
- magicalDamageTaken
- physicalDamageTaken
- trueDamageTaken
- goldEarned
- goldSpent
- turretKills
- inhibitorKills
- totalMinionsKilled
- neutralMinionsKilled
- neutralMinionsKilledTeamJungle
- neutralMinionsKilledEnemyJungle
- totalTimeCrowdControlDealt
- champLevelvisionWardsBoughtInGame
- sightWardsBoughtInGamewardsPlaced
- wardsKilled
- firstBloodKill
- firstBloodAssist
- firstTowerKill
- firstTowerAssist
- combatPlayerScore
- objectivePlayerScore
- totalPlayerScore
- totalScoreRank

References:

Kenneth Hall. 2018. LoL-Match-Prediction. [online] Available at: < https://github.com/minihat/LoL-Match-Prediction> [Accessed 22 May 2020]

pseudonym117. 2019. *Welcome to RiotWatchers Documentation! – RiotWatcher 3.0.0 documentation*. [online] Available at: < https://riot-watcher.readthedocs.io/en/latest/> [Accessed 22 May 2020]

League of Legends Wiki. 2020. *Nexus*. [online] Available at: <https://leagueoflegends.fandom.com/wiki/Nexus > [Accessed 22 May 2020].

League of Legends Wiki. 2020. *Summoners Rift*. [online] Available at: <<https://leagueoflegends.fandom.com/wiki/Summoners_Rift> [Accessed 22 May 2020].