Predicting League of Legends Matches

Benson Chen and Kevin Lei

University of Pennsylvania

1 Introduction

League of Legends is a multiplayer online battle arena (MOBA) game created by the video game publisher, Riot Games. Although there are many different game modes, the most common type is a 5v5 where players are separated into two teams of five. A players is known as a summoner and each summoner controls a champion which has a set of unique abilities and attributes. Before the start of the game, each team chooses their champions and other settings. The goal is for the teams to battle each other using their champions, using items and experience that are obtained in-game. The games currency is gold which can be obtained from killing enemy champions or killing third party characters. There are also a variety of structures assigned to each team, one of them being the nexus. The game ends when one team is able to successfully destroy the other teams nexus.

LoL is on the forefront of the Esports scene along with other games such as Dota and Starcraft.

Every year, LoL hosts an world championship tournament. In 2015, the LoL world championship was held in several locations, from Berlin to Paris. In 2015, the average game had a concurrent viewership of 4.2 million unique viewers. To put that to perspective, the average episode of season 5 of Game of Thrones has around 7-8 million viewers. The prize pool for the 2015 LoL world champions amounted to over 2 million USD.

2 Goals

This project has two main goals:

1. Create a model that predicts the probability of win given match statistics.
2. Find out what variables contribute most to winning a game, so that we can create strategies that will give better chances of winning the game.

For the first goal, although it may not make much sense to use the match information to predict the probability of winning, because the match will have already happened by then, there are other uses. One is that this model can be a good proxy for predicting the probability of winning when the game is still happening. League of Legends is a heavily broadcasted Esport, and having a measure of how the game is doing is an important metric.

For the second goal, we really want to look at what variables most indicative of a winning match-up. Using this information, we can construct strategies that can give teams a sense of what they should strive to do during the game to get an advantage.

3 Data Collection

The first step in our project was collecting the data from Riots League of Legends API. We used two of the endpoints: (1) match-v2.2; (2) matchlist-v2.2. The first endpoint, match-v2.2, returns information related to a given match. Specifically, it takes a matchId (unique identifier for each match) as a parameter and returns a data structure containing statistics relating to the participants, teams, and events within the game. The second endpoint, matchlist-v2.2, returns match information for a given summoner (someone who plays LoL). The endpoint takes a summonerId (unique identifier for each summoner) and returns a list of matches that the summoner has played. We narrowed down our data set to ranked 5v5 games played in 2015 season.

To accomplish this task, we wrote a Java program which uses these two endpoints to acquire a list of summoners and then subsequently a list of matches.

To acquire a list of summoners, we used two methods: getMatches and getSummoners. The first method, getMatches, takes a summonerId and calls the matchlist-v2.2 endpoint to obtain all of the matches that the summoner has played. This list then gets written to a file. The second method, getSummoners, takes matchId and uses the match-v2.2 endpoint to obtain the list of summonerIds which had participated in that given match. Using these two methods in conjunction allowed us to randomly obtain a large number of matches. Firstly, we picked a random seed summonerId. We passed this into getMatches and chose a random match to obtain new summoners. We constructed two queues to hold a list of potential summonerIds and matchIds and continuously called the two methods in a loop, using getMatches to add more matchIds to the match queue and using getSummoners to add more summonerIds to the summoner queue. If we imagine a graph of summonerIds where the vertices are matchIds and edges connect matches with the same participants, we traversed the graph using a Depth First Search (DFS) to obtain our list of matches.

After obtaining our list of matches, we used a different method, populateMatches, which took in a matchId and called the match-v2.2 endpoint to obtain features for each match. The method parsed the JSON output to obtain information about the champions used in each match, gold received by each team during a given time frame, wards and items bought and used, and other features relevant to a teams success. We have outlined a more specific description of the features in the Data Overview section below.

4 Data Cleaning

After obtaining our raw data from our Java program, our next step was to clean the data. Our first concern was that Riots API did not always return well-formed JSON. This resulted in some of the features containing null values or empty strings. Since we could always use our program to obtain more matches, we simply removed any matches with these malformed features. This left us with a cleaned dataset of 3,000 matches.

We also removed any features from our dataset that were not useful. For example, we had indicator variables for each championId. In particular, championId of 420 corresponded to Illaoi, a champion released about a month ago for the 2016 season. Since we had only collected data for matches in the 2015 season, we never saw this champion used and subsequently removed its corresponding indicator variable.

5 Data Overview

In this section, we summarize our data and give descriptions for our features.

5.1 Sample Size

We use a sample of 3,000 random matches of LoL games.

5.2 Response Variable

Our response variable is categorical 1-0 variable that is 1 if team 1 wins, and 0 if team 2 wins.

5.3 Champions

The first and obvious feature that we included was the champions themselves. Excluding the newest champion Illaoi, for which we do not have data, there are 127 total champions. To simplify our task, we approached this feature in the following manner:

Champion is on team 1	Assigned a value of 1
Champion is on team 2	Assigned a value of -1
Champion is not in current game	Assigned a value of 0

We could have used separate indicator variables for champions being on teams 1 and 2, but we wanted to limit the size of our feature space in order to guarantee that our methods ran fast enough.

We see from the following table the summary statistics of the frequency of champions played (on either team):

Minimum	1st Qt.	Median	Mean	3rd Qt.	Maximum
14.0	83.5	159.0	236.2	350.5	1166.0

6 Works Cited

http://www.lolesports.com/en_US/all-star/articles/worlds-2015-viewership
https://en.wikipedia.org/wiki/Game_of_Thrones
http://loldevelopers.de.vu/
https://developer.riotgames.com/api/
http://www.nytimes.com/2014/08/26/technology/amazon-nears-a-deal-for-twitch.html