Global Power Rankings Hackathon

Author: Daniel Yao
Email: devdaniely@gmail.com

API Examples

/global_rankings?number_of_teams={int}

- https://mqb2k0rcn5.execute-api.us-west-2.amazonaws.com/Prod/global_rankings
- https://mqb2k0rcn5.execute-api.us-west-2.amazonaws.com/Prod/global_rankings?number_of_teams=50

/tournament_rankings/{tournament_id}?stage={stage_name}

API Gateway (30sec timeout)

- https://mqb2k0rcn5.execute-api.us-west-2.amazonaws.com/Prod/tournament_rankings/105873410870441926?stage=round_1

Lambda Invoke (5min timeout)

- https://hpbgvizwe3dxaiydlvjh5y3xli0vymmz.lambda-url.us-west-2.on.aws/tournament_rankings/105873410870441926?stage=round_1

/team_rankings?team_ids={team_ids_array}

API Gateway (30sec timeout)

- https://mqb2k0rcn5.execute-api.us-west-2.amazonaws.com/Prod/team_rankings?team_ids=98926509885559666&team_ids=98767991877340524

Lambda Invoke (5min timeout)

- https://cg25zzqwk2unw5xq3c67riazz40ztpmt.lambda-url.us-west-2.on.aws/team_rankings?team_ids=98926509885559666,98767991877340524,98767991892579754,98767991853197861

Formula

$$Rating_{new} = Rating_{old} + K(GameResult - (Expected_{win} + WinPredictRatioDiff))$$

- $K = 15$
- $GameResult = 1\ or\ 0\ \ (win/loss)$
- $Expected_{win} = \frac{1}{1+10^{\frac{(ProximitySum_{teamA} - ProximitySum_{teamB})}{400}}}$

****

- $WinPredictRatioDiff = abs(WinPredictRatio_{teamA} - WinPredictRatio_{teamB})$
- $WinPredictRatio = \frac{actualGoldSpent^{1.83}}{actualGoldSpent^{1.83} + totalGold^{1.83}}$

ProximitySum is the average number of times 3 players on a team were in close proximity. Close proximity is defined by this dev post from Riot Games

ActualGoldSpent is gold spent by the team minus gold that was passively earned. In other words, actualGoldSpent is the amount of gold spent using ONLY the bonus income that was earned through objectives.

Rating_old: Each team starts with an initial rating of 1200

Example

Team	Win	totalGold	earnedGold	goldSpent	actualGoldSpent	proximitySum
TL	1	50211	34043	42818	26650	17.14955
DIG	0	39018	22850	37235	21157	13.10038

Expect = 1 / (1 + 10 ^ ((17.141955 - 13.10038) / 400)) = **0.49**

DiffWinPredictRatio = 0.098 - 0.066 = **0.032**

- TL = 26650^1.83 / (26650^1.83 + 89229^1.83) = 0.098
- DIG = 21157^1.83 / (21157^1.83 + 89229^1.83) = 0.066

Rating: 1200 + 15 * (1 - (**0.49** + **0.032**))

- TL Rating = 1207.103
- DIG Rating = 1192.103

Methodology

The main ELO rating formula shown here is derived from Chess, Baseball, and Basketball sports analytics. The constant values were borrowed from a wiki that describes the ELO ratings formula used in League of Legends. However, the 2 main variables of this formula that require some clarification are the WinPredictRatio and ProximitySum.

WinPredictRatio

The WinPredictRatio is based off the Pythagorean Expectation formula. The Pythagorean Expectation is used to evaluate over/under performing teams by predicting their win percentage. Typically, this would require a scoring system that exists in zero-sum games, but since the gains and losses throughout a League of Legends game are not strictly balanced, gold efficiency is used instead.

Gold efficiency is an indicator of how well a team is managing their resources. At the end of the game, comparing how gold is managed by each team provides a stronger insight into the team's collaboration. The team's kill/death/assist (KDA) ratio may not provide as much insight as gold since they are actually ways of generating bonus gold income.

With this in mind, the WinPredictRatio uses the **actualGoldSpent**, which is the amount of gold spent using ONLY the bonus income that was earned through objectives.

ProximitySum

ProximitySum is another measurement used to evaluate performance at the team level. When players on the same team are within a close proximity, they have a better chance of making a play. This becomes an indicator of team coordination and cohesive strategizing as they have a better chance of responding to events. However, because a common strategy in League of Legends is "Split Pushing", where a player is separated from the rest of the team to exert pressure across the map, proximity has a slightly different definition here.

Referencing a dev post by Riot, close proximity is considered when players are within a range of 2000 units. Using the positional data provided, we can count the number triangles formed by the players.

Proximity is the number of triangles formed by the positions of the 5 players **with an area less than the 2000 unit range circle.** As seen in Figure 1, the pink triangle formed by 3 players has an area less than the range circle. To account for Split Pushing and other cross map strategies, the green triangle will also have an area less than the range circle and their proximity included. **ProximitySum** is the average sum of all the triangles that meet this criteria over the duration of the game.

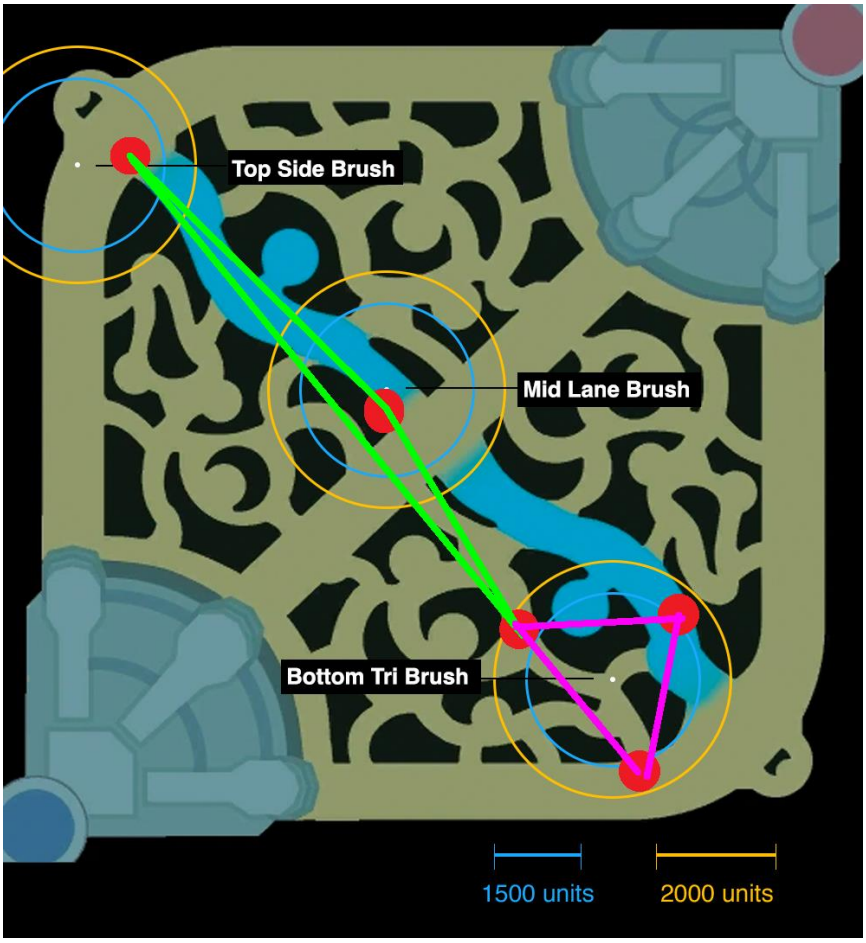


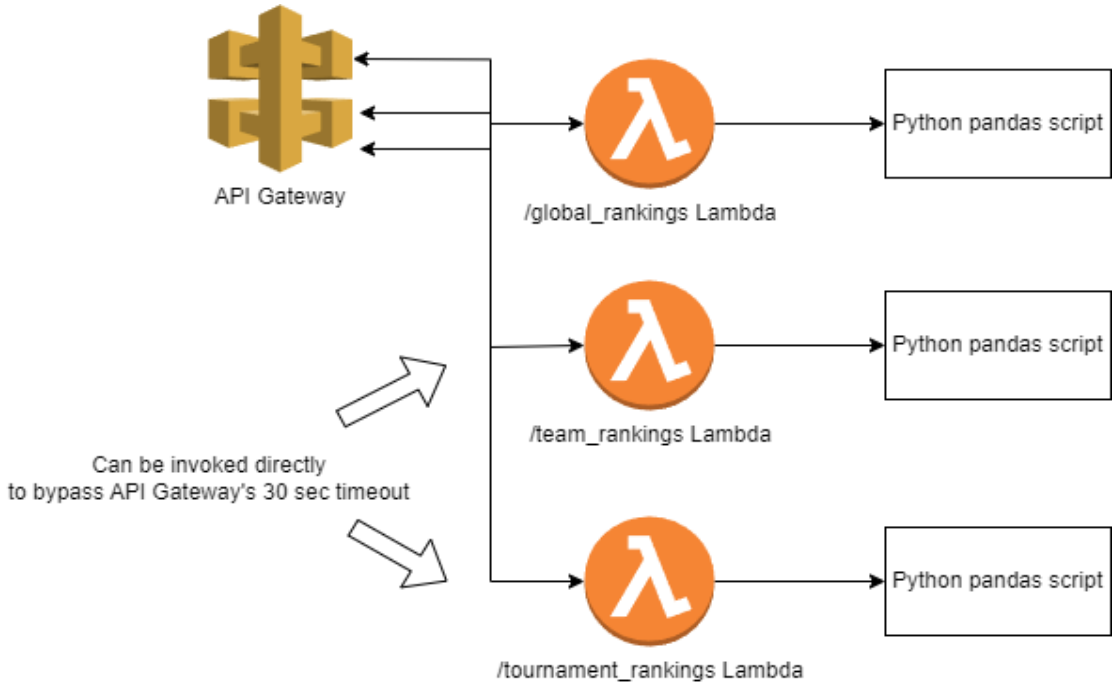
Figure 1: Red Dots are positions of players on the same team

For every combination of 3 for the 5 player positions (5 Choose 3 = 10 possible triangles), the triangle areas were compared to the radius of the 2000 unit proximity.

- Proximity Circle = πr^2 where r = 1000 (half of the 2000 unit range)

- Triangle Area = $\frac{1}{2}bh$

Design



Future Work

Future work includes adjusting the weighting of the formula to improve accuracy. There are also more variables that can be extracted from the positional data provided. Another area I would've liked to explore is the sequencing of events that occurred in the game. Using DNA sequencing algorithms, we could extract common subsequences to evaluate different strategies teams use.

Resources

Data/Code

- Github Code: https://github.com/devdaniely/lolanalysis-hackathon
- Oracle's Elixir Endgame Data: https://oracleselixir.com/stats/teams/byTournament

Research Links:

- Calculating ELO: https://leagueoflegends.fandom.com/wiki/Elo_rating_system#Calculating_Elo
- Getting Proximity: https://lolesports.com/article/dev-diary-changes-to-proximity/bltc57ec217dbf2a162
- https://www.doranslab.gg/articles/location-based-champ-metrics.html
- NCAA Basketball Ranking Formula: https://towardsdatascience.com/college-basketballs-net-rankings-explained-25faa0ce71ed
- KenPom Basketball: https://kenpom.com/blog/ratings-explanation/
- Pythagorean Expectation: https://en.wikipedia.org/wiki/Pythagorean_expectation