# LeagueNet

**Sergio Sardar and Brandon Walker**
Department of Computer Science and BioEngineering
Stanford University
ssardar@stanford.edu
bmwalk@stanford.edu

## Abstract

League Of Legends (LoL) is a game with 100 million monthly active players. Many players craft ever-evolving strategies on how to win and climb the ranked ladder. A tool to evaluate the likelihood of winning a match based on certain variables would be invaluable to both professional organizations who compete for multi-million dollar prize pools and casual players who want an edge. We aimed to create a model that intakes variables such as player ranks and champions in game to output whether a team would win or lose. The optimal model we found to work was a four layer deep neural network with a batch normalization layer and ReLu activation function at the output layer. We achieved an accuracy of 62 percent on the test set.

## 1 Introduction

As mentioned in the abstract, a tool to evaluate the likelihood of winning a match based on certain variables would be invaluable to both professional organizations who compete for multi-million dollar prize pools and casual players who want an edge over competitors. In a league of legends game, 5 players play together against 5 others. Each player picks a unique character, or champion. There are benefits and drawbacks to choosing each champion; some synergize well with others while picking the wrong champion could make the game significantly harder. Our model aims to predict the output of the game based on pre-game variables. While we had a glut of potential input features to choose from, our experimentation showed that the best indicator of success before the game has even started are the ranks of each player in the game and the champions the players picked. Rank correlates to the skill of each player. For example, the rank 1 player in North America is better than everyone else in the region. Our input vector had 20 values: 5 ranks for one team, 5 ranks for the other, and 10 more integers that represented the champion each player chose. So an input vector looks something like:

'rank of player 1, rank of player 2...rank of player 10, champion1...champion10'

team 1 corresponds to the first five ranks in the input vector while team 2 corresponds to the next 5 ranks in the input vector.

The output is simply a 1 (corresponding to a win for team 1 ) and a 0 (corresponding to a loss for team 1).

## 2 Related work

This is actually the first piece of work done in predicting game results (both traditional sports and e sports) based on pre-game knowledge. There are two papers that loosely resemble the work that we are doing, but there are still significant differences. The first paper is the only other paper that involves deep learning and MOBAs (Multiplayer online Battle Arenas) called "StarCraft II: A New Challenge for Reinforcement Learning". However, the goal of this research project was to teach a

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

system to play the game rather than predict it. The other paper is called, "Football Match Prediction using Deep Learning" by Daniel Pettersson and Robert Nyquist, which attempted to predict the result of soccer games using a recurrent neural network and data points taken at regular intervals during a match. However, a key difference here is that their approach relies on data taken during our match while we intend to solely depend on data we can gather before the match has even started. As a result, there was no baseline architecture we could really refer to for some inspiration: a recurrent neural network like the soccer prediction model didn't make sense as we weren't dealing with any form of sequenced data.

## 3 Dataset and Features

The data collection was perhaps the most difficult aspect of this experiment. To train the model, we had to train it on thousands of match histories that weren't publicly available. The first step of the data collection process was figuring out how to find the usernames of the top 1000 players in each major region that plays League Of Legends, North America, Europe and Korea. We managed to find this on an obscure Korean website that manually tracks League of Legend statistics called op.gg. We wrote a webscraper to manually scrape the usernames off of the website which lacks an API. After we managed to gather 3000 usernames, we were then able to query the Riot APIs in a multi-step process that involved around 20,000 API calls over two weeks in order to get 60,000 match results. (Riot is the company that develops League Of Legends, but unfortunately is stingy with the amount of data they make available to developers). Player rank, however, is not a statistic that is officially tracked by Riot. To collect this data, we had to write another webscraper that queried op.gg and another website called LoLKing to extract the rank of each player that appeared in a singular match result data point. Tragically, the rank of some players that appeared in these match results were unavailable on these sites so we ended up with 8441 match result data points where we had the rank of each of the 10 players who appeared in each match. This was simply not good enough. As a result, we had to write scripts to augment the data by swapping the first 5 ranks in an input vector with the second 5 ranks, followed by flipping the 1 or 0 that the input vector would map to.

After augmentation, we ended up with 16882 training examples, and decided to split the data into 90/5/5 chunks for train/dev/test sets respectively.

## 4 Methods

We're using a four layer, batch normalized, fully connected neural net with the output layer using a ReLu activation function. While we weren't taught how to use other architectures that are suited to a classification problem like this in class, we attempted to use a restricted boltzmann machine as another model.

The model that ended up performing better, the four layer neural network,ended up using an optimizer and a sparse softmax cross entripy loss function. Several configurations, such as using momentum for the optimizer, were attempted to be incorporated into the model, but the configuration as we will submit with the code worked the best.

## 5 Experiments/Results/Discussion

The hyper paramters were as follows:

learning rate: 1e-2, batch size: 64, dropout rate: 0.3, number of neurons per layer: 30.

The hyperparamters were determined after a few weeks of experimentation.

Here's a table of the results for the two models we tested:

| | Training Accuracy | Test Accuracy | Train set size | Dev set size |
|---|---|---|---|---|
| 4 layer deep neural net | 0.57 | 0.62 | 15999 | 883 |
| Restricted Boltzmann Machine | 0.56 | 0.58 | 15999 | 883 |

Table 1: Results

We believed we were initially over-fitting our data due to a high variance issue we were encountering but managed to substantially bring that down using dropout with a rate of 0.3.

## 6 Conclusion/Future Work

We ended up finishing this project with the a 62 percent accuracy which is pretty good considering the fact that League of Legends is a game that is highly influenced by small mistakes in the game, and we were predicting results based on only information that one can know before entering a game itself.

Since no one has ever tried to conduct a study like this on League of Legends, with the most similar study being a paper on predicting soccer results using recurrent neural networks, we weren't sure what to expect. A lot of the architectures we studied in class, such as RNNs and CNNs, didn't really make sense because of the nature of the data and desired output, so a lot of research and experimentation was needed to find an appropriate neural net. We're sure the results of the net could be better. With the comprehensiveness of the data we gathered from Riot and op.gg, there are so many more input features we could add. However, we would need to spend a great amount of time in collecting more data and tweaking the architecture accordingly. Had we come into this project with a lot more data, we are positive the results would have been substantially better.

## 7 Contributions

Sergio developed the webscrapers, conducted the majority of the data collection and augmentation, set up the input pipeline and fleshed out the model. He also conducted the model training, hyperparameter tweaking, designed the presentation session poster and wrote the writeup.

Brandon helped develop the concept of the project, collected some of the data required for the project and helped presenting it.

## References

LINK TO CODE: https://drive.google.com/open?id=1i6X7vpJtDEGpIvDhEXuq1klkCfwN727l

[1] Pettersson, Daniel, and Robert Nyquist. "Football Match Prediction Using Deep Learning." 2017. Accessed March 22, 2018. http://publications.lib.chalmers.se/records/fulltext/250411/250411.pdf.

[2] StarCraft II: A New Challenge for Reinforcement Learning. Accessed March 23, 2018.

[3] Tensorflow

[4]BeautifulSoup4