Dota2 Pick/Ban Recommendation System

Vhitesh More
COMP student, dept. Computer Engineering
VCET Mumbai, India

Atul Naik
COMP student, dept. Computer Engineering
VCET Mumbai, India

Juneeth Panjri
COMP student, dept. Computer Engineering
VCET Mumbai, India

***Abstract*— Multiplayer Online Battle Arena (MOBA) games have received enormous recognition recently. In a match of such video games, players compete in groups of 5 every controlling an in-recreation avatar, known as heroes, selected from a roster of more than a hundred. The choice of heroes, additionally known as select or draft, takes place earlier to the match and alternates among the two teams till each participant has selected one hero. Heroes are designed with one of a kind strengths and weaknesses to promote group cooperation in a sport. Intuitively, heroes in a sturdy group need to supplement each different strength's and suppress those of combatants. Hero drafting is therefore a tough hassle due to the complicated hero-to-hero relationships to consider. In this paper, we recommend a singular hero recommendation device that indicates heroes to add to a present group at the same time as maximizing the crew's prospect for victory. For that case, we represent the drafting among teams as a combinatorial sport. Our empirical assessment indicates that hero groups drafted by our recommendation set of rules have substantially better win chance in opposition to groups constructed by using other baseline and modern-day strategies.**

Keywords—Logistic Regression, K-Nearest Neighbors, Defense of the Ancients

I. INTRODUCTION

Dota 2 by Valve is a popular computer game with a player base of 8 million unique users and increasing. While many of these users play just for fun, there are professional players that take part in Dota 2 competitions to win massive monetary payouts. For example, Valve's recent Dota 2 tournament, "The International 9," had a prize pool of over $34 million.

In every Dota2 match being played two teams each of 5 players fight against each other. At the beginning of any match, every individual player has to choose an in-game character know as 'hero'. For any hero selected by the player, no other player can select that same hero for in that match. Heroes have a broad pool of mechanics and spells/abilities that, added with the massive hero pool, makes each match unique. A fascinating aspect of the game is that in selecting a hero, players must remember not only the individual strengths and shortcomings of each hero, but also how the strengths and weaknesses of that hero interact with the heroes already chosen by other players. An effective hero pick is one that synergizes with the heroes chosen by teammates, and both

exploits the shortcomings and decreases the strengths of the heroes chosen by the opposing team. Assuming equally skilled teams, the aftermath of hero selection can be so astonishing that well devised hero choices can totally give a team a large advantage before the match even begins. The goal of our project is to recommend heroes that will perform well against an opposing team of heroes.

This is a fitting as a classic machine learning recommendation problem, but what intrigues us is the sheer volume of the solution space. With 119 heroes to select from and five heroes per team, we are trying to find the best five heroes for any given matchup, which results in over eight quadrillion possible team combinations. On a deeper level, recommending heroes with the help of machine learning is difficult because it tries to apprehend via raw data what professional players have devised a gut instinct for through hundreds of thousands of hours of game play time.

Every hero has different strengths and weaknesses. The term farm means to collect gold and experience. These strengths and weaknesses may complement other heroes on the friendly team or counter heroes on the opposing team. Generally, hero roles can be segregated into the following types:

- Carry – A type of hero that depends a lot on items and experience becoming stronger late game.
- Support – This hero helps to get the carry all the farm and takes very less farm himself.
- Initiator – This hero has the ability to initiate team fights that favour his team.
- Ganker - This hero has the ability to rotate onthe map and disrupt the opponent's carry's farm. Any team not having any of the above-

mentioned roles will definitely have a bad time in the match.
There appears to be two fascinating machine learning problems identifying with a Dota 2 match. First is hero recommendation, suggesting a hero at each step of the drafting process to maximize the probability of winning. Second is win rate prediction, forecasting which team will be victorious based on some given match data. The first problem is very dependent on the second apparently. The success of a hero recommendation system depends on accurate win predictor.

II. DATASET

We used Valve's Steam web API to pull data for 56691 matches between June 5 and September 7. Our data satisfies the following requirements:

1) The game has different modes
   i. all pick,
   ii. single draft
   iii. all random
   iv. random draft
   v. captain's draft
   vi. captain's mode
   vii. least played
2) The modes have been designed by keeping in mind the true vision of DOTA, we can choose any hero to play a match.
3) Only 8% of players correspond to a skill level very high. We believe utilizing only very-high skill level matches allows us to best represent heroes at their full potential.
4) Only those, matches are considered in which no player leaves the match before it ends. Thus, the System doesn't capture how absent players or heroes are affecting the outcome of the match.

The data for each match has been structured as JSON and includes the heroes that were chosen by both teams, how the heroes and players performed over the course of the game, and who won the match ultimately. The JSON data for each match has been stored in a MongoDB database during data collection.

We exported 77% of the matches from our database to form a training set of 63,797 matches. We exported the remaining 7.3% of our database to form a test set of 7,421 matches.

III. DRAFTING PHASE

5) Roles and team composition

1. Cores : As per our knowledge of the game , we think that Mid, Carry and Offlane can be considered the Cores of the team. we'll talk about offlane later, but a better game strategy is one in which mid and safelane complement one another. One of the heroes plays the role of building spells and items to help deal magical damage in the late game, and the other deals physical damage. In absence of any of them it, would be easier for the enemy to build items. Eg. Again, if in case, you have TA mid, you don't really need Sven in the safe lane, though he's a strong hero and would solo against few offlaners. In this case Morphling or Necrophos may complement TA well by dealing with great magical burst damage.

2. Supports : Supports can be categorized in more ways is what I think this could be because my natural game role is support and it is where I have more experience. One support is greedy (position 4) and then there is sacrificial (position 5). The position 4 focuses on the way in which his items would help change the course of the game, this are core heroes or simply crucial, like blink dagger, as the hero initiates fights with ganked disables. The position 5 buys sentries, dusts, smokes, wards and is usually satisfied with wand + boots even in the mid game. Meaning, one support has defensive buffs/heals/saves/ such as movement speed or armour to save and protect crucial heroes from direct damage and also sometimes prevent hard lockdowns from the enemy. The other support is disables/initiates/ offensive and may possess a key high damage skill at times, that is specifically strong in the early stage of the game. Eg. Like Oracle and Vengeful spirit when play the game role as supports are usually defensive and use their ultimates when the core gets into a hard lockdown, such as Primal Roar and Lasso. While heroes such as, Shadow Shaman and Lion primarily have heavy disables and burst damage.

3. Offlane : The offlane position can be considered as most flexible, versatile, when it is about their role in that particular game. Traditionally, their role has been to initiate the battles with enemy and disable a hero, but their game role has evolved over time. The offlane heroes are said, to fill a gap that the team composition doesn't have. This is not mutually exclusive, but the most popular/strongest offlaners are those that can: who can march solo into the enemy safe lane, therefore being able to not only contest the enemy carries farm, but also, get farm and solo xp. Retreat from enemy jungle whenever we, face the need, when there is Dota2 Pick/Ban Recommendation System

pressure from the enemy support team. Could be facing a hard Lockdown i.e. a spell or ability that pierces spell immunity (BKB and the like), or even better, which we cannot dispell or counter with a Linkens Sphere. Make space for the team, by either being extremely elusive, powerful and hard to kill or simply have controlled units that would push through the lane. eg. Dark Seer and Nature's Prophet who will split push, when there is nothing else with the team.

6) Counters and team synergy

General rules: A simple but not strict guideline that has been traditionally applied to Dota from the very beginning can be applied to most high-level games.

Split push helps to counter team-fight and heavy lineups.

Ganking/map control would help counter with Split push lineups.

Team-fight/death-ball will help to counter ganking/map control lineups.

algorithm, he states that Dota2cp models hero selection as a zero-sum-game for which it learns the game matrix by logistic regression. When suggesting hero picks and bans the system considers the teams as min-max agents that pick hero one at a time in turns. While Dota2cp is a great first effort at a hero recommendation engine, we believe we can improve on its results by exploring other machine learning algorithms.

V. METHODOLOGY

A. Feature Vector

In Dota 2 matches, the teams are referred as 'radiant' and 'dire'. These terms are roughly analogous to 'home' and 0,

metrics
are hero advantages (matchup between two heroes) i.e. the advantage one hero holds over another and win
rates i.e. the no. of matches won vs. the no. of matches played. Unfortunately, the creators have not divulged specific
details of their approach nor their accuracy.
DotaBuff[2] is a website from which we can acquire various stats of Dota2 . At present, the site does not provide any type of win prediction or hero recommendation system, but the data Visualizations that they provide are really useful.
Dota2cp [3] is a web application developed with a similar aim of recommending heroes for Dota 2 matches. If there are two teams consisting of five heroes each, the author reports a 63% accuracy of rightly predicting the team that would win the match. Though, the author does not release source code for his

'away', as this only determines the start point of both the teams on the game world map, which is roughly symmetric.
Dota2 consists of 119 heroes (as of writing), but the web API uses hero ID numbers which range from IDs 1 to 129 (some hero IDs are not used), so for the algorithms that
we use a feature vector has been used $x \in \mathbb{R}$ such that:

$$x_i = \begin{cases} 1 & \text{if a radiant player played hero with id i} \\ 0, & \text{otherwise} \end{cases}$$

$$x_{119+i} = \begin{cases} 1 & \text{if a dire player played hero with id i} \\ 0, & \text{otherwise} \end{cases}$$

We also defined our label $y \in R$ to be:

$$y = \begin{cases} 1 & \text{if radiant team won} \\ 0, & \text{otherwise} \end{cases}$$

B. Making Predictions

As our dataset contains the information about heroes on teams in the specific radiant vs. dire configurations, full data is not utilized by simply running the algorithm by on data of every match match in the dataset.
Instead, we use the following procedure to make predictions. Given a feature vector, which we call radiant_query:

1) Get the probability that team radiant in the query wins, by using the algorithm.

2) Design Dire_Query by interchanging Dire and Radiant teams in the present query, the bottom half of the feature vector and top half are radiant and dire team respectively.

3) Get probability that dire team wins by using the algorithm on dire query, the probability that the radiant team in radiant_query would lose the match in-case it would have been the dire team instead.

4) Compute overall probability of the match overall_pro is computed as the average of rad_pro and (1 - dire_pro):

overall_pro = (rad_pro+(1-dire_pro)) / 2

5) Predict the outcome of the match according w_i 216

to the radiant_query as such the radiant team is winning if overall_prob > 0.5 and as the dire team is winning otherwise.

This procedure takes into consideration, matches in our dataset that might not have the team configuration of a given query in a single direction (e.g. radiant vs. dire) but may have the configuration in the other direction (e.g. dire vs. radiant).

C. Logistic Regression

Logistic regression is an algorithm model that predicts a binary output i.e. yes or no only two outputs (it chooses between two options) by using weighted sum of predictor variables. At, first we trained a simple model of logistic regression with an intercept term for predicting the outcome of a match. The feature vector and label that we have used is described in part A of this section.

1) Learning Curve: A plot of our learning curve for logistic regression is made. The training and test accuracies are quite close together, indicating that our model does not overfit the data. The test accuracy of our model asymptotically approaches 69.8% at about an 18,000 training set size, indicating that 18,000 matches is an optimal training set size for our logistic regression model.

2) Analysis: We can see that logistic regression model proves that hero selection alone is an important indicator and affects the outcome of a Dota 2 match. However, since logistic regression model works by purely considering a weighted sum of our feature vector (which only indicate which heroes are there on either of the teams), it fails to capture the synergistic and antagonistic relationships that exist between heroes.

D. K-Nearest Neighbor

K-nearest neighbor algorithm is a non-parametric method for classification and regression models that predict objects'

And class memberships based on the k-closest training model examples in the feature space. We have designed a custom weight and distance function and chose to utilize all the training examples as "nearest neighbors." Our polynomial weight function that is described below aggressively gives less weightage to dissimilar training examples. The feature vector and label that we used have been described in part A of this section.

1) Calculating Weights: The following equation represents a combined distance and weighting function used in our K-nearest neighbors simulations:

$$ {}^{(i)\ d} \left(\qquad \right) = \frac{\sum_{j=1} \text{AND}(q_i, x_j)}{\text{NUM\_IN\_QUERY}} $$

The d parameter represents the polynomial scaling of the weight function. x represents the feature vector for a training match i and is compared by using logical AND operator with a query vector q. j is the hero ID of each respective vector. NUM_IN_QUERY is the total no. of heroes present in the query vector the function has been normalized to in between 0 and 1, and it gives more weight to matches that resemble query match more closely. For this, the a function is designed which compares the query match vector with every training match vector that is present in the dataset and counts the instances in which the hero is present in both the vectors. A larger d will result in similar matches getting much more weight than dissimilar matches.

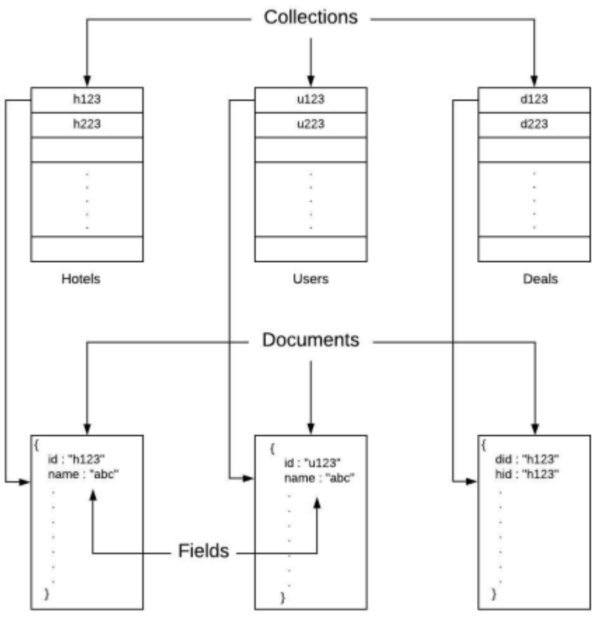
2) Choosing an Optimal Weight Dimension: For choosing the optimal d dimension parameter that has been described above, we have used k- fold cross validation method with k = 2 on 20,000 matches from our training set and have varied d across otherwise identical K-nearest neighbors models.

3) Learning Curve: A plot of our learning curve for K-nearest neighbors is shown in Figure 3. The test accuracy of four model monotonically increases with training set size up to nearly 70% for around 50,000 training matches. As, we do not see the learning curve level off, more data may help to improve the accuracy.

E. Implementations

For the backbone of the project MongoDB is used. MongoDB is a high-performance NoSQL database, built around the JSON data format and it also provides dynamic data. It includes project data structure. Each database has its own files in the file system with multiple databases located on the same MongoDB server. A collection of database documents can be referred to as a collection. RDBMS that is equivalent to a collection could be considered as a table. The entire collection exists within a single database. In mongo DB we have three collections hotels, users and deals. Within the collection, different documents cover different fields. A set of key-value pairs can be created as a document.

The collection document contains a variety of data fields.



We have downloaded the matches from a specific id as per the time frame according to the game patch we want to consider for training. By using the steam API we downloaded matches with only ‘high’ and ‘very high’ skill. Only the matches which are full length that is no player has left the match are being considered. We have collected up-to 500 matches in one go for every 15 minutes.

1) Recommendation Engine Architecture

Our system, is designed using greedy search as it considers all the heroes that could be added to the team also, it is programmed to rank the candidate by probability of winning against the enemy team if the candidate was a part of the team.

We, have designed a modular system any of the given algorithms could be used to recommend given, ID numbers of the heroes on both teams, the recommendation engine proceeds as follows:

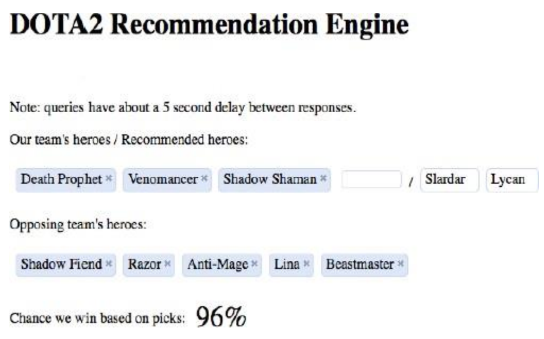
a. A feature vector is created of the match as described, (Part A. Methodology)

b. Create a set of new feature vectors covering every possible vector, each with a different candidate hero added to the original feature Vector.

c. Run the algorithm to compute the probability of victory with the set of feature vectors from step 2, averaging the probability that the team wins in both radiant and dire configurations of the game as described in subsection 2.

d. Sort the candidates by probability of victory to give
Ranked recommendations.

2) Web Interface



DOTA2 Recommendation Engine

Note: queries have about a 5 second delay between responses.

Our team's heroes / Recommended heroes:

Death Prophet × Venomancer × Shadow Shaman × / Slardar Lycan

Opposing team's heroes:

Shadow Fiend × Razor × Anti-Mage × Lina × Beastmaster ×

Chance we win based on picks: 96%

The data entry interface has been adapted from the Dota2cp website To facilitate the use of our recommendation engine for users so that the user can easily input the heroes by typing the first few letters of a hero’s name, choosing the desired hero from a list of auto-complete suggestions, and pressing enter. We have connected this web interface to our recommendation engine using the Flask Python web application library.

REFERENCES

[1] http://en.wikipedia.org/wiki/Defense of the Ancients
[2] Dota2 Picker. http//www.dotapicker.com/
[3] Dota Buff. http//www.dotabuff.com/
[4] K. Conley and D. Perry, “How does he saw me? a recommendation engine for picking heroes in dota 2,”CS229 Previous Projects, 2013.
[5] https://developer.valvesoftware.com/wiki/SteamWebAPI
[6] http://wiki.teamliquid.net/dota2/The International/2019
[7] http://dev.dota2.com/showthread.php?t=58317
[8] Peter Harrington, “Machine Learning In Action – Chp5.Logistic Regression”, Manning Publications 2012.
[9] The Medium. https://medium.com/capital-one-tech/k-nearest-neighbors-knn-algorithm-for-machine-learning-e883219c8f26/