Dota2 Heroes Recommendation System

Kaisong Fan
ECEN Department
Texas A&M University
kfan@tamu.edu

Zhe Chen
ECEN Department
Texas A&M University
chenzhe95123@tamu.edu

Bohuai Jiang
ECEN Department
Texas A&M University
bohuaijiang.1021@tamu.edu

Siyuan Sun
ECEN Department
Texas A&M University
sxs144730@tamu.edu

ABSTRACT

In this paper, we present a Dota2 hero recommendation system. We first discuss related works and compare them with our own approaches. We develop two different neural network, SoftMax and Sigmoid approach, to do the hero recommendations. We then introduce different evaluation criteria for these two approaches. We also introduce the system architecture about how our system works.

KEYWORDS

Neural Network; Recommendation; Dota2; Machine Learning;

1 INTRODUCTION

The Dota2 is a multiplayer online battle arena (MOBA) game developed by Valve Corporation. Doat2 has millions of players and becoming more and more popular. There are many Dota2 tournaments held every year. For example, The International 2018's total prize pool reaches $25,532,177. The hero combination of a team is of vital importance. Even though the skill levels for two teams are the same, the team with poor hero combinations will easily lose the match.

In our project, we built a hero recommendation system that recommend heroes to players to form a good team combination. Users choose four teammates and five opponents and our recommendation engin will recommend top five heroes that will give users the best chance to win the match.

We developed two different neural network models, SoftMax and Sigmoid model, to do the recommendation respectively. Both of them behaves well under corresponding evaluations. We also developed a well-designed web interface for players to use.

2 RELATED WORK

Word2Vec used neural network approach to construct a multiclass classification model which has the same idea as SoftMax model except the Word2Vec's input is 1-hot vector. The Word2Vec model used negative sampling to reduce high computation, but since SoftMax does not have a large feature size, it will not need Negative Sampling.

How Does He Saw Me? is a Dota2 recommendation engine that trained model using Logistic Regression and K-nearest Neighbors. In this paper, the authors states that Logistic Regression fails to capture the synergistic and antagonistic relationships between heroes, however by using K-nearest Neighbors and their own custom weight, the model is able to achieve nearly 70%.

3 METHODOLOGYY

In order to build a Dota2 heroes recommendation, our project used the classical feed-forward neural network to learn from the dataset. The purpose of neural network model is to learn different heroes' formation on each team and based on the match result, our model could predict the best hero to help our team to win. During the developing stage, we implemented two type of models Sigmoid and SoftMax, and each model has a different approach of learning dataset.

3.1 SIGMOID MODEL

3.1.1 Model Description. The goal of this model is to predict the winning chance of "Radiant" team based on the heroes somposition. Sigmoid model used "Adagrad" optimizer and "binary crossentropy" loss function. The last output layer used 'Sigmoid' activation function and its outputs from a range 0-1 which is the probability of winning.

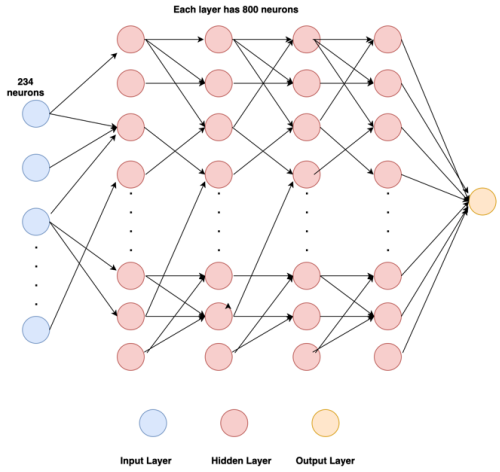

Figure 1: Sigmoid Model Structure Overview

In order to find the optimal neural network structure, we trained our model over all possible numbers of hidden layers and neurons. The Figure 2 below shows the prediction accuracy which been tested using validation dataset.

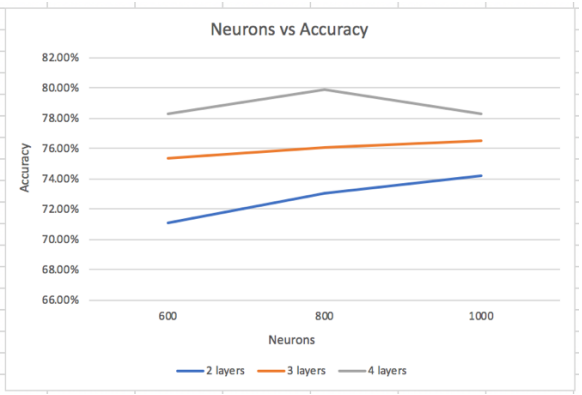

Figure 2: Neurons vs Accuracy

3.1.2 *Training Dataset.* In the Dota2 matches, there are 117 heroes have been used, and each game match has a "Radiant" and "Dire" team. We labeled each hero in range of 0-116.

$$\boldsymbol{X}_i = \begin{cases} 1, & Radiant's\ heroes\ selection\ with\ hero\ ID\ i \\ 0, & Otherwise \end{cases}$$

$$\boldsymbol{X}_{i+117} = \begin{cases} 1, & Dire's\ heroes\ selection\ with\ hero\ ID\ i \\ 0, & Otherwise \end{cases}$$

For the label of each match, the label will be 1 if "Radiant" team won and 0 otherwise.

$$Y = \begin{cases} 1, & if\ Radian\ team\ won \\ 0, & Otherwise \end{cases}$$

The input vector contains two teams' hero selection which is $|X| = 234$, and we could utilize our dataset by swapping top and bottom vectors to generate a new match. The label will also change to the opposite of original match. By using this method, we are able to achieve over 120,000 matches data.

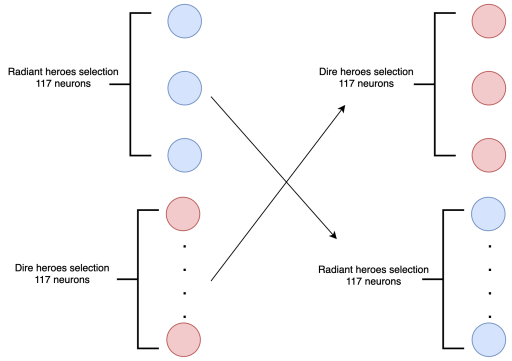

Figure 3: Training Data Expansion

3.1.3 Prediction. Since users have to choose 9 unique heroes, 4 in our team and 5 in opponent team, and the model predicts top 5 recommendation to a user.

Assume P denotes a set of unselected heroes which $size(P)$ equals to 108, and during the prediction, the model predicts the winning chance for each unselected hero and returns the top 5 highest scores. Below is the pseudocode.

For each i in $\{P\}$ *do*
    $\boldsymbol{X}[i] = 1$
    $Reseult.append[\ Predict(\boldsymbol{X})\ ]$
    $\boldsymbol{X} = original(\boldsymbol{X})$

Return Result$[Top\ 5\ ]$

3.2 SOFTMAX MODEL

3.2.1 Model Description. The SoftMax model is a simple multiclass classification, the input vector size is still $|X| = 234$ but the output layer contains 117 neurons each represents a hero.

The model used 'Adam' optimizer and "categorical crossentropy" loss function. The last output layer used 'SoftMax' activation function so that each neuron has a percentage.

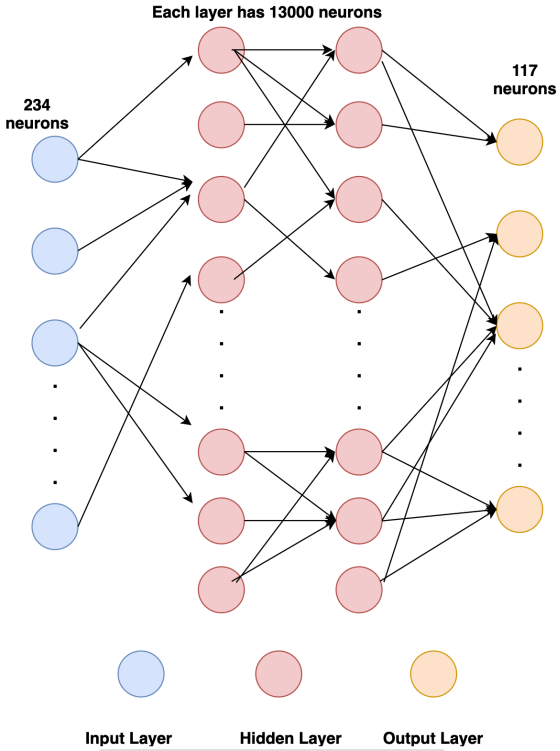

Figure 4: SoftMax Model Structure Overview

We also optimized model structure using the validation dataset, and the optimal solution is two hidden layer each contains 13000 neurons.

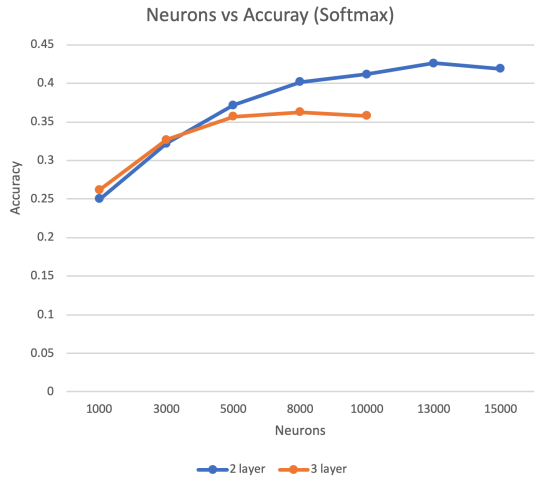

Figure 5: Neurons vs Accuracy

3.2.2 *Training Dataset.* In this model, instead of using "Radiant" and "Dire" team, the input vector forms by winning and losing team. There are four heroes will be used in the winning team and the 5th hero will be used in label vector Y which is a 1-hot vector.

$$\boldsymbol{X}_i = \begin{cases} 1, & Choose\ 4\ heroes\ from\ winning\ team\ with\ ID\ i \\ 0, & Otherwise \end{cases}$$

$$\boldsymbol{X}_{i+117} = \begin{cases} 1, & Losing\ team's\ heroes\ with\ hero\ ID\ i \\ 0, & Otherwise \end{cases}$$

$$\boldsymbol{Y}_i = \begin{cases} 1, & One\ hero\ from\ winning\ team\ with\ ID\ i \\ 0, & Otherwise \end{cases}$$

To utilize the dataset, each match can split into 5 matches by using 5 different heroes from the winning team as the label vector, and that is over 300,000 matches data. This method provides more data for model to train and allows the model to learn the differences of each hero in a similar team formation.

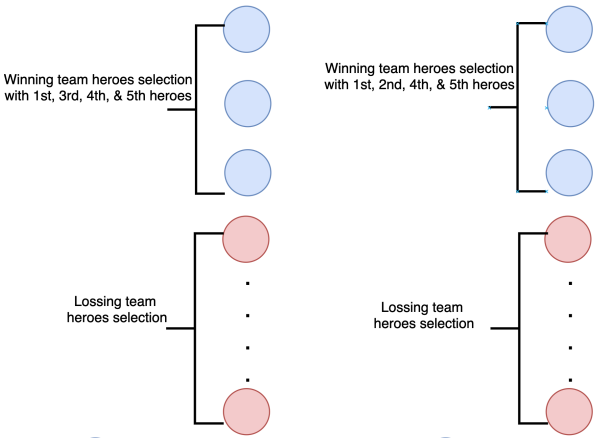

Figure 6: Training Data Expansion

3.2.3 *Prediction.* The prediction is very simple for this model, once the user chooses 9 unique heroes, 4 in our team and 5 in opponent team, the model will assign each hero a probability and the heroes with the highest probability will be recommended.

4 TRANING AND EVALUATION

In this part, we will show loss during both training and validation stages. After expanding the datasets, both Sigmoid and SoftMax model have over 200,000 and 300,000 data respectively. The dataset was then shuffled and split into 80% for training and 20% for validating.

4.1 Sigmoid Training. After training the model for 8 epochs, the loss is almost zero, and it means the model learns little after that. For the training process, it needs about 8 epochs to make the result relatively accurate.

Figure 7: Training Loss vs Epochs

4.1.2 Sigmoid Validation. In order to avoid overfitting, the model was tested by validation dataset using formula below. For every match, the model predicts twice by swapping top and bottom vectors of X. The prediction of $P(Opponents\_Teammates)$ is the chance that "Teammates" losing the game.

$$Winning\ chance = \frac{P(Teammates\_Opponents) + (1 - P(Opponents\_Teammates))}{2}$$

After 10 epochs, the validation accuracy saturated and the highest accuracy is about 80%.

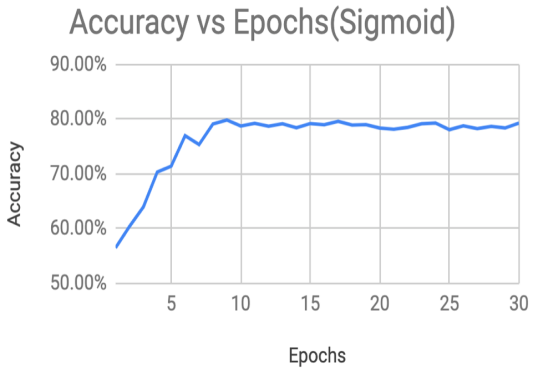

Figure 8: Validation Accuracy vs Epochs

4.2.1 SoftMax Training. The model used here is a 2 layer, 13000 neurons per layer model. For the training loss, the loss function is "binary crossentropy". From the graph, the loss value drops rapidly before 5 epochs and stays stable after 6 epochs.

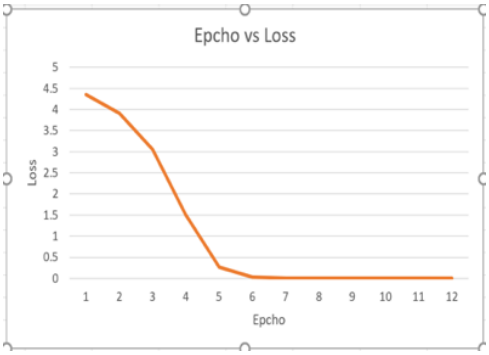

Figure 9: Training Loss vs Epochs

4.2.2 SoftMax Validation. During the validation stage, a formula was also designed, and the purpose of this formula is to calculate the percentage of actual hero shows up in top 5 recommendation using over 50,000 validation datasets.

$$\text{count} = \begin{cases} count + 1, & actual\ hero \in Top5\ recommendation \\ count + 0, & Otherwise \end{cases}$$

$$percentage = \frac{count}{total\ number\ of\ matches}$$

From the figure below, the accuracy increases in the first 8 epochs, and slowly decreases after 8 epochs. The highest accuracy is 52% after 8 epochs.

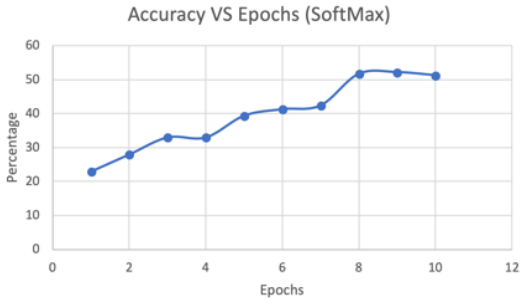

Figure 10: Validation Accuracy vs Epochs

5 OVERVIEW

5.1 System Architecture. The frontend is developed with React and Node.js, and the backend server is developed with Flask. Every time it receives a http GET request with parameters of heroes' id as well as the chosen model, it will execute the corresponding pre-trained model, SoftMax or Sigmoid model, to get the top five recommended heroes and eventually send the result back to the frontend.

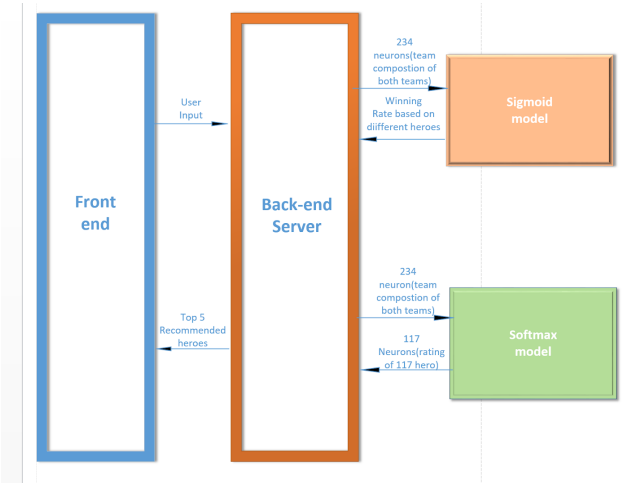

Figure 11: System Architecture

5.2 User Interface. The web UI let users choose their teammates' heroes and opponents' heroes via dragging the images. Once four teammates' heroes and five opponents' heroes have been chosen, users can click button "SoftMax" or "Sigmoid" to choose the model they want to use. Once the button is clicked, the chosen model will predict the result and display on the webpage.

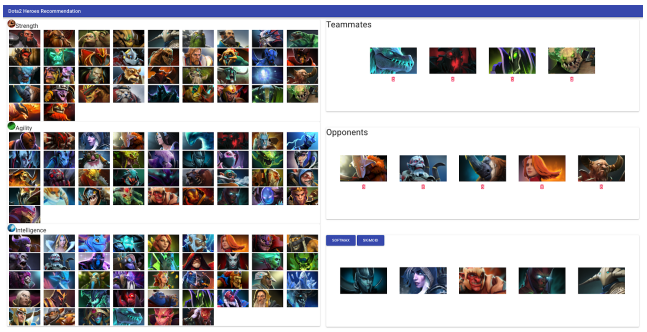

Figure 12: Web UI

6 CONCLUSION

In this paper, the recommendation engine has two approaches Multiclass Classification and Logistic Regression. By using the neural network technique, both models can recommend heroes to users, however, these models also appear to have a significant difference due to the design and structure.

For Sigmoid model, it was trained based on Radiant team's heroes and Dire team's heroes formation to find the winning chance. The model then predicts unselected heroes and return the recommended heroes that have the highest probability. Since the model does not directly do the recommendation, it would actually increase some errors. Also, Sigmoid model only learns the heroes composition so it cannot capture the synergistic and antagonistic relationships between heroes.

On the other hand, SoftMax model utilized the dataset by splitting each match into 5 matches. It learns the pattern of all five heroes against opponents' team. It is a surprise that the SoftMax model requires such large hidden layers and neurons to achieve its optimal. It might due to the large output layer, and in order to be optimized, the model requires a complex hidden layer to learn the dataset.

There have been many challenges when designing the models, such as overfitting using built-in loss function and constructing input vector. In our final design, the input vector has size of 234 and contains two teams instead of using a 117 vector for two teams. This structure helps the model to learn the differences of team composition and it actually improved the performance. The other challenge was the overfitting issue during validation. After research and study of the dataset, we realized that if the model predicted hero A when actual hero was B, it does not mean the prediction, hero A, is wrong because some heroes are similar and could be the second choice. After implementing our own evaluation formula, the result shows a reasonable accuracy. To get a better evaluation, a professional player could be the best human judgment since the relevance of the result could vary based on player's preference and experience.

7 FUTURE WORK

The recommendation system can only recommend 1 hero when other 9 heroes are given. We can modify the system to recommend two heroes for a 3vs4 team formation, but it will require to design and train a different model.

In the training process, if the model's prediction differs from the actual result, the loss function will calculate a high penalty. In this scenario, some heroes that have a similar skill set would not be a wrong prediction, so a customized loss function for both models can improve the accuracy and performance.

REFERENCES

[1] Chris McCormick. "Word2Vec Tutorial – The Skip-Gram Model". Mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/.

[2] Conley Kevin and Daniel Perry. "How Does He Saw Me? A Recommendation Engine for Picking Heroes in dota2"