Using LSTM and AAE to Predict Dota2 Outcomes by analyzing the Draft

Jian Zhao
School of Software Engineering, Shanghai Jiao Tong University
atthebrink@sjtu.edu.cn

Abstract

Dota2 is a very popular online game around the world, which not only attracts millions of players and hundreds of clubs by its highest championship prizes in the game industry, but also gets attention from academic groups like OpenAI recently because of the complex strategies and competitive play styles in this game. However, no advanced machine learning algorithms are designed to forecast the winning probability by analyzing the draft. To address the problem, we proposed a network using LSTMs combined with DNN layers, which can learn the temporal dependencies of hero choices and deep latent relationships between them. And we processed datasets using robust anomaly detection methods based on AAE networks. With these efforts combined together, the model yields an accuracy of 61.3% ,which is higher than the traditional methods up to 3 percent, and an AUC score of 0.651.

1 Introduction

Defense of the Ancients 2 (Dota2) is a multiplayer online game, which is developed by Valve. Dota has been a very popular video game around the world for a very long time. As the sequel to Dota, Dota2 swept the world with more than 50 million players and fans. And there are hundreds of clubs and professional players compete in its global professional championship which is called the International. The International holds every year with prizes sponsored by Valve and the fans all over the world, resulting in the highest prizes among all video game championships. However, the development of deep learning and other advanced machine learning algorithms sheds little light on electronic sports field by now.

If we can predict the outcome of a Dota2 game using advanced machine learning algorithms, we can help professional players and clubs in many ways. And this prediction of the outcomes can also act as a base stone of training deep reinforcement learning agent to plan strategies before the game. The OpenAI team, who has noticed the great research potential in Dota2, have done a research about training multiple agents playing against human professional players in Dota2 and have achieved a very good result using advanced deep reinforcement learning. Although OpenAI shows that AI agents powered by deep reinforcement learning can beat top professional human players in Dota2, there are still limitations and huge gaps between AIs and humans. Currently, OpenAI need to limit the heroes in Dota2 that can be chosen and the choice of the heroes is just hard-coded. To make AIs truly skilled and competitive as human professional players, they need to learn how to pick certain heroes and plan strategies that give them the best advantages. There is still a long way towards this goal. And the first step may be to predict the outcomes of a match.

There are two teams, known as Radiant and Dire, that respectively consists of five players in each game. Each player controls one of 114 heroes (by 2018), which are unique avatars of the players and can only be picked once. In a Dota2 match, before the main game starts, players in each team will firstly pick their heroes by turns. In professional match, there will also be a ban move, which bans players from picking certain heroes. This ban-picking stage (or draft stage) becomes a game to collect those heroes that have great synergy effect and avoid your opponents picking heroes that counters your picks. In Dota2, heroes have relative synergistic and antagonistic relationships, these relationships can affect the win rate to some extent. And almost all the professional players pays much attention to the draft stage, because they have experienced failure due to bad ban-picks. Unfortunately, there is no reliable way for players to evaluate there choice of ban-picks before the match for now. and the goal of this paper is to propose an algorithm that can reliably predict the outcome of a match from the ban-picks.

We first define the problem we want to solve and discuss how can we solve it. We define the combination of the two teams' picked heroes as a draft $D = \{h_i | h_i \in H, i = 0, 1, ..., 10\}$ where H is the set of all available heroes.(see Fig.1) And we recognize the problem as to find a mapping that maps D to its wining probability $P(D)$. According to the Principle of Permutation and Combination, the number of possible draft D is

$$\binom{114}{5} \times \binom{114-5}{5} \times \frac{1}{2} \approx 8.6 \times 10^{15}$$

Due to the huge scale of D, we cannot simply iterate on the space D and calculate the winning probability table $P(D)$.

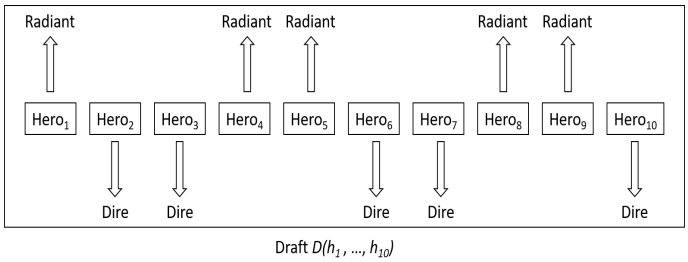

Figure 1: Definition of Draft Stage, where Radiant and Dire pick heroes according to pre-defined game rules (bans are omitted).

So we need to extract the latent features of the certain combinations and use these features to predict the probability of wining, and neural networks perform well in learning latent features and predicting.

What is to be discussed next is whether the problem should be a classification problem or a regression problem. Since it is ether Radiant wins or Dire wins, we can predict the probability of Radiant wining or the probability of D belongs to class *Radiant Win*. First, if we treat this problem as a regression problem, then our model will output a consistent probability representing the chance of Radiant wins. To achieve this, we will need to know some true probability as the teaching signal. But it is difficult to know the winning probability or to use the frequency as an estimation. In fact, we collected over 1,300,000 matching records and found no same D in them. We cannot apply supervised learning because the observation $O(D) = Radiant$ will not prove $P(D) = 1$ and the observation $O(D) = Dire$ will not prove $P(D) = 0$. Thus we believe it is difficult to model this problem as a regression problem.

For the reasons mentioned above, we argue that this problem should be a classification problem, but there are still challenges need to be addressed. We defined two class *Radiant Win* and *Dire Win* as the target classes and trained a densely connected neural network classifier. The classifier failed to distinguish the two class well as the accuracy can not reach 60%. We believe the reason that causes poor performance lies deep in the data itself. Suppose the true probability of Radiant wins under draft D is $P(D)$ and we collect an observation $O(D)$, then $P(O(D) = Radiant)$ is $P(D)$ and $P(O(D) = Dire)$ is $1 - P(D)$. In our classification, we define $P(D) > 0.5$ as class *Radiant Win*, but we know that there may be observations that violate this criteria, e.g. $P(D) = 0.7$ but we observe Dire won. This means that there are noises in the data set or we can say that some data may not be very *representative* especially when $P(D)$ is close to 0.5 and there are half data expected to have a wrong label.

To address the issue, we propose a method that drop the *unrepresentative* samples using robust approach based on AAE(adversarial autoncoder), which extract the latent features of D and map it to a Gaussian Distribution to detect data anomalies and iterate on the data set by retraining the model using adjusted sample weights. By using the clarified data set, we got performance gains and the final accuracy of the algorithm reached 0.613[1].

2 Related Work

Although OpenAI achieved great success in training AIs to be smart enough to beat professional human players, Dota2 got attention from researchers only recently. First articles were mostly descriptive, general and theoretical, investigating, for example, rules and fair play maintenance of the games [Conley and Perry, 2013] or correlation of leadership styles of players with roles in the game (carry, support, jungler etc.) they choose to play [Nuangjumnonga and Mitomo, 2012]. In the first quantitative research of Dota 2, authors analyzed cooperation within teams, national compositions of players, role distribution of heroes and some other stats based on information from its web forums [Pobiedina *et al.*, 2013].

Later researchers discovered the potential of the data provided by the game itself soon after that and started using to test hypothesis, detect patterns and make predictions. For example Rioult et al. [Rioult *et al.*, 2014] analyzed topological patterns of DotA teams based on area, inertia, diameter, distance and other features derived from their positions and movements of the players around the map to identify which of them are related with winning or loosing the game. Drachen et al. used Neural Networks and Genetic Algorithms to analyze and optimize patterns of heroes movements on the map in DotA [Drachen *et al.*, 2014]. Eggert et al. [Eggert *et al.*, 2015] applied classification algorithms to heroes game statistics to identify their roles. However most researched topic was not win prediction from heroe drafts.

Conley and Perry were the first to demonstrate the importance of information from draft stage of the game with Logistic Regression and k-Nearest Neighbors (kNN) [Conley and Perry, 2013]. They got 69.8% test accuracy on 18,000 training dataset for Logistic Regression, but it could not capture the synergistic and antagonistic relationships between heroes inside and between teams. To address that issue authors used kNN with custom weights for neighbors and distance metrics with 2-fold cross-validation on 20,000 matches to choose d dimension parameter for kNN. For optimal d-dimension = 4 they got 67.43% accuracy on cross-validation and 70% accuracy on 50,000 test datasets. Based on that results authors built a recommendation engine with web interface. However one of its drawbacks was it's slow speed: for k=5 kNN took 4 h and 12 h for cross-validation.

Although their work was the first to show the importance of draft alone, the interaction among heroes within and between teams were hard to capture with such a simplistic approach. Agarwala and Pearce tried to take that into account including the interactions among heroes into the logistic regression model [Agarwala, 2014]. To define a role of each hero and model their interactions they used PCA analysis of the heroes' statistics (kills, deaths, gold per minute etc.). However, their results showed inefficiency of such approach, because it got them only 57% accuracy while the model without interactions got 62% accuracy. But its worth noticing

[1]Code is available at https://github.com/atthebrink/Dota2-Outcomes-Prediction

that although the PCA-based models couldn't match predictive accuracy of logistic regression, the composition of teams they suggested looked more balanced and reasonable from the game's point of view. Another caveat of their approach was that they took data from different sources: the data on match statistics was taken from public games while stats on heroes were based on professional games. This might bias the results because public games are completely different from professional ones and match stats from the former should not be mixed with heroes stats from the letter. In short, they didn't use heroes roles directly and replaced them with PCA components to model the balance of teams. Besides that, they tried to find some meaningful strategies with K-Means clustering on end-game statistics but couldn't find clusters which means that no patterns of gameplay could be detected on their data.

Another approach to that problem of modeling heroes' interactions was proposed by Song et al. [Song *et al.*, 2015]. They took 6,000 matches and manually added 50 combinations of 2 heroes to the features set and used forward stepwise regression for feature selection. They chose data from the "All Pick", "Ranked All Pick" and "Random Draft" without leavers and zero kills. 10-fold CV logreg: 3,000 matches total: 2,700 vs. 300. Training error 28%, test error – 46% . They concluded that only addition of particular heroes improves the model while the others might cause the prediction go wrong.

Kalyanaraman was the first one to implicitly introduced the roles of the heroes as a feature in the model of win prediction [Kalyanaraman, 2015]. Author took 30,426 matches from the "All Pick", "Random Draft", "Single Draft", "All Random", "Least Played" and "Captain Draft" game types because in theory it should represent all the heroes in the best way since appearance of any particular hero depends on game type. They filtered the matches by MMR to select only skilled players and took the games which were at least 900 s and used ensemble of Genetic Algorithms and logistic regression on 220 matches. Logistic regression alone return 69.42% and ensemble with Genetic Algorithm and logistic regression approached 74.1% accuracy on the test set. Although it's the highest result among all the articles in the review, lack of ROC AUC information and small sample of matches, chosen for the Genetic Algorithm, hampers its reliability.

Another attempt to include interaction among heroes was done by Kinkade and Lim, who took 62,000 matches with "very high" skill level without leavers and game duration at least 10 min [Kinkade, 2015]. 52,000 training, 5,000 testing and 5,000 validation. Tried Logistic Regression and Random Forest with such feature of a pairwise winrate for Radiant and Dire. The feature could capture such relationships as matchup, synergy and countering and each of them increased the quality of the model up to 72.9%. Logistic Regression and Random Forest on picks data only. Got 72.9% test accuracy for Logistic Regression and overfitted Random Forest which gave them after tuning only 67% test accuracy. It is worth mentioning that their baseline, which included highest combined individual win rate for the heroes, had 63% accuracy.

Some authors expanded the scope of win prediction from draft information to other data from the game. Johansson and Wikstrom wrote a thesis where they trained Random Forest on the information from the game (such as amount of gold for each hero, his kills, deaths assists for each minute etc.) which had 82.23% accuracy at the five minute point [Johansson and Wikström, 2015]. Although such accuracy seem to be very high, that fact that it's based on data from the game events makes its use very limited, because it demand real-time data to be practically useful.

2.1 Contribution

From the previous research we've found the following shortcomings:(1)vague data acquisition strategies (for example, its not clear why authors filter players by their skill level and only use games with high MMR);(2)small or incorrect samples of data (sometimes data sets were gathered during periods when some changes in the game mechanics were introduced or the samples were mere thousands of matches);(3)inappropriate assumptions: the relationship between heroes does not independently affect the win rate, treat this influence as independent event will cause error in prediction.

Hence our contribution is:(1)using properly mined data set containing match records of all range of skill levels; (2)using large enough data[1] (1397634 records) to generate reliable results; (3)using deep learning methods to automatically extract the relationships between heroes and using LSTM to extract additional information in the picking sequence to get better accuary. (4)using AAE to robustly detect unrepresentative data and exclude them form future training.

3 Method

The methods used in this experiment can be summarized into two steps: We want to exclude the unrepresentative data, so the first step is data processing. After that, we fit a LSTM model on the processed data set and evaluate our results using reliable metrics. We will first introduce our data set and then illustrate these two steps in detail.

3.1 Data Set

We used a data set collected directly from Valve's Dota2 game servers using the APIs they provided. This data is collected by Andrei Apostoae in 2017 and it is collected when the game version is stable. The original data contains 1397634 match records that include which team won, the indexes of heroes being chosen, the game modes and average MMR (representing player skill level). These records covers all range of MMR but mainly includes three mods: ALL Pick, which permits players to pick any hero and is the most common game mode; Random Draft, which randomly select 50 heroes for players to choose; and Captain Mode, which includes ban moves and is the formal modes for professional games.

We do not exclude any data in the data set based on game modes or MMR, because we believe that the latent synergy or countering relationship between heroes should be universal. We use the data set that covers all MMR range and game mods to avoid overfiting on features that are local to some game modes or some MMR range. we used two representations of hero drafts as input of the algorithms. First is just

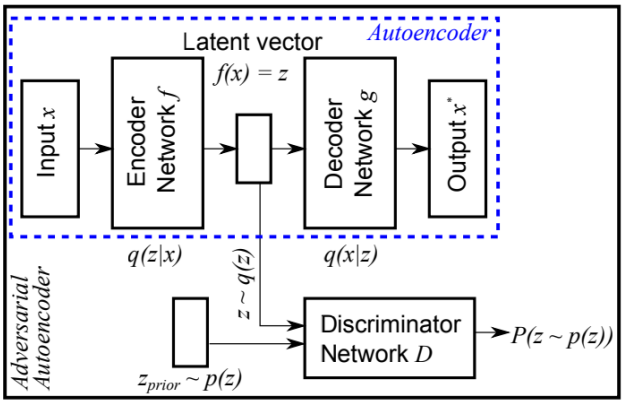

Figure 2: Schematic structure of conventional autoencoder (blue dashed box) and the extension to an adversarial autoencoder.

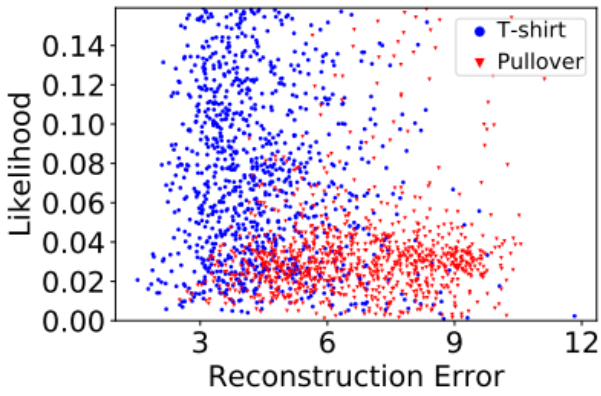

Figure 3: Reconstruction error and likelihood for an AAE trained on a clean subset of Fashion-MNIST containing only class 'T-shirt' (blue). Test data from the anomaly class 'Pullover' (red) yield lower likelihood values and higher reconstruction errors.

"bag of heroes" technique, where each draft is encoded as a binary vector of length $2 \times N$ where N is the size of hero pool with

$$x_i = \begin{cases} 1, & \text{if } i \leq N \text{ and hero } i \text{ was in the radiant team} \\ & \text{or if } i > N \text{ and hero } i - N \text{ was in the dire team} \\ 0, & \text{otherwise} \end{cases}$$

We use this draft encode as the input of densely connected neural networks as this technique of encoding is often used in DNN classifier. The second is "sequence of picking", where each draft is encoded as a 10-element vector containing hero indexes with

$$x_i = n,\ 0 \leq n \leq N \text{ and } N \text{ is the size of hero pool}$$

These indexes are sorted in the sequence that the heroes are picked, thus we can use this encoding format as the input to LSTM networks to gain additional temporal information.

As mentioned above, the data sets we collected are just observations of outcomes, they may not truly reflect the probability of Radiant or Dire wining. Indeed, we expected a certain proportion of incorrectly labeled data. Also, for some records, the true win rate may be very close to 50%, which means that these records cannot reflect latent relationships very well and the observations may be more random. We recognize these samples as unrepresentative samples, if the model overfits on these unrepresentative samples, it can not learn the latent synergy or countering relationships well.

3.2 Data Processing

To address this issue, we used adversarial autoencoder. Adversarial autoencoders (AAE)[Makhzani *et al.*, 2015] extend the concept of autoencoders by inducing a prior distribution p(z) in the latent space. A generative model of the data distribution pdata(x) is thus obtained by applying the decoder to samples from the imposed prior in latent space. The main difference to Variational autoencoders[Kingma and Welling, 2014] is the uses the recently proposed generative adversarial networks (GAN)[Goodfellow *et al.*, 2014] to perform variational inference by matching the aggregated posterior of the hidden code. As a result, AAEs can impose any prior distribution from which samples can be drawn, and have smoothly varying outputs in data space for small changes in corresponding latent space. An example of an AAE structure is displayed in Fig. 2.

Let x be the input and z be the latent code vector (hidden units) of an autoencoder with a deep encoder and decoder. Let $p(z)$ be the prior distribution we want to impose on the codes, $q(z|x)$ be an encoding distribution and $p(x|z)$ be the decoding distribution. Also let $pd(x)$ be the data distribution, and $p(x)$ be the model distribution. The encoding function of the autoencoder $q(z|x)$ defines an aggregated posterior distribution of $q(z)$ on the hidden code vector of the autoencoder as follows:

$$q(z) = \int_x q(z|x)p_d(x)dx$$

Both, the adversarial network and the autoencoder are trained jointly with SGD in two phases – the reconstruction phase and the regularization phase – executed on each mini-batch. In the reconstruction phase, the autoencoder updates the encoder and the decoder to minimize the reconstruction error of the inputs. In the regularization phase, the adversarial network first updates its discriminative network 2 to tell apart the true samples (generated using the prior) from the generated samples (the hidden codes computed by the autoencoder). The adversarial network then updates its generator (which is also the encoder of the autoencoder) to confuse the discriminative network. Once the training procedure is done, the decoder of the autoencoder will define a generative model that maps the imposed prior of $p(z)$ to the data distribution.

From the perspective of anomaly detection AAEs are interesting because apart from the reconstruction error, the latent code provides an additional indication for anomalies [Leveau and Joly, 2017]. Simply put, we expect anomalies x (characterized by low $p_{data}(x)$) to map to latent representations with low density $p(z|x)$, or otherwise have high reconstruction error $L(x, x_0)$, because high likelihood latent should be decoded into normal input (see Fig. 3).

We use a method proposed in [Beggel *et al.*, 2019]. Since

AAEs impose a prior distribution $p(z)$ on the latent representations z, the likelihood $p(\hat{z})$ under the prior of a new code vector $\hat{z} = f(\hat{x})$ can be used as an anomaly score [Leveau and Joly, 2017]. Anomalies are expected to have lower scores than normal examples. However, it is also clear that $p(\hat{z})$ alone is an imperfect score, because anomalies in local clusters with small support might indeed be assigned higher scores than normal examples in boundary regions. Furthermore, the encoder might not be able to learn a mapping that exactly reproduces the prior. Despite these weaknesses, a likelihood-based criterion is able to identify most anomalies with similar performance as a reconstruction-based approach, and in addition allows a combination of both approaches. We split our methods into four phases and they are described as follows:

1. **Pre-training:** First, we need to initialize the AAE but we cannot train it with the complete data set, because this will let the AAE learn the features from normal data as well as unrepresentative or wrongly labeled data. Thus we first train a shallow neural network with two densely connected layers of 128 neurons on the full data set for a fixed epochs (5 in this experiment). Due to the shallowness and small number of training epochs, this network is less likely to overfit on the data set. Indeed, this DNN gives a 60% accuracy on both train set and test set (during our standalone test). Although this DNN is not well trained to classify the wins and losses, we believe it captures the most representative features to some extent. We feed the data set to the DNN and use the output probability of the correct class as advantages by assigning each data sample with a weight $w_i = y_{Radiant}$ if Radiant won or else $w_i = y_{Dire}$, where $y_{Radiant}$ and y_{Dire} are the output of the DNN. This is the starting point of training and refining the AAE.
2. **Initialization:** The AAE is trained on the complete data set with weights initialized in phase 1 for a fixed number of epochs (3000 in this experiment). To avoid overfitting the AAE, we uniformly sample from the complete data set each time to built a mini-batch. After this phase, the autoencoder and the latent code generator-discriminator are both converged and then we can start our refinement on the weights.
3. **Refinement:** We feed the complete data set to the AAE and record the reconstruct error E and likelihood score L. Then we sort the two arrays in descending and ascending order respectively and truncate these two arrays to the first αN elements, where α is a constant expected anomaly rate (0.4 in our experiment), to get the candidate anomalies denoted $\hat{E}$ and $\hat{L}$. All elements within $\hat{A} = \hat{E} \cap \hat{L}$ are assigned a new weight $w_i = 0$, thereby being removed from further training and refining.
4. **Re-training:** After we detecting anomalies in the training set, we take advantage of these candidates to re-train our model such that the reconstruction error on anomalies increase and likelihood score on anomalies decrease. To do so, we assign temporally weight $w_i = -1$ for $x_i \in \hat{A}$ and re-train the model for a small group of mini-batches (50 in our experiment and these mini-batches are still uniformly sampled from $]hatA$). After this re-training, the model is forced to learn a higher reconstruction error and lower likelihood for anomaly candidates. Although this method is expected to erroneously reconstruct the false negatives worse, it can still reduce the true anomalies in the training set. Phase 3 and 4 are repeated for d times (5 in our experiment) where the model iteratively exclude anomaly candidates and refine itself.

We apply this data processing method on the complete data set (1397634) and finally get a cleared data set containing 745201 samples. And the accrucy and loss curves of the AAE model we used in this method is shown in Fig 4.

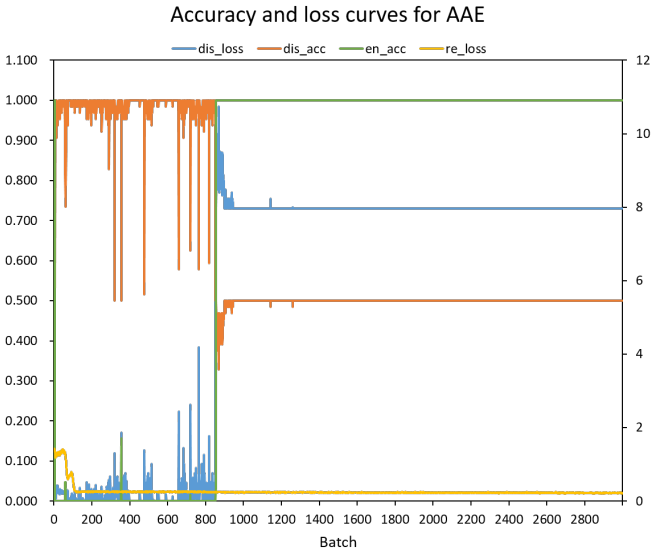

Figure 4: dis_acc: accuracy of the discriminator which tries to distinguish fake latent codes from real prior distribution; en_acc: accuracy of encoder which tries to fool discriminator with fake codes; dis_loss: loss of discriminator; re_loss: reconstruction loss. dis_loss is marked on the right axis while others are marked on the left axis.

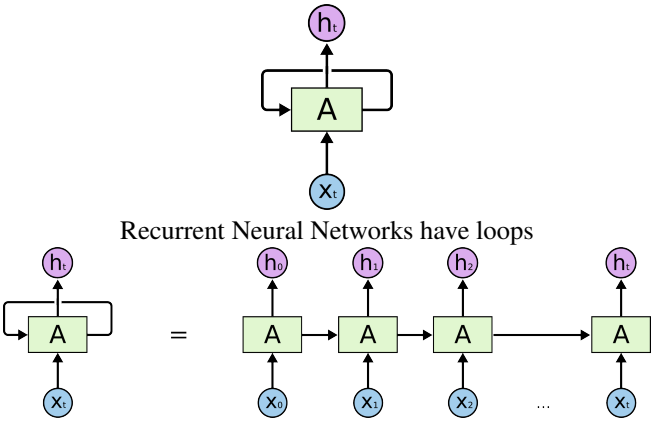

Figure 5: An unrolled recurrent neural network

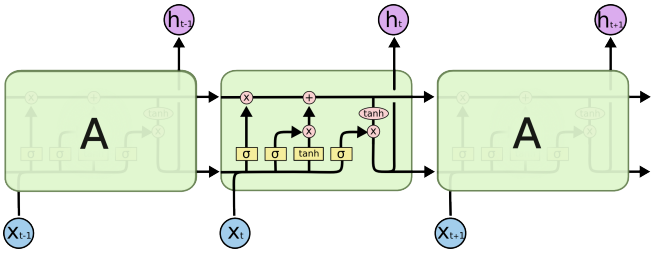

Figure 6: Typical structure of a LSTM neuron.

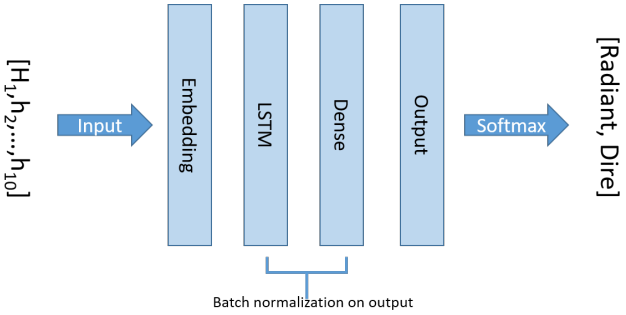

Figure 7: Architecture of network

3.3 Win Rate prediction

After processing data with AAE, we expect the unrepresentative or wrongly labeled data is reduced in the data set. Since the draft D is naturally a sequence of decisions with a temporal characteristics (see Fig 1) and people often believe that the last few decisions are the most important ones, A recurrent neural network is the best choice to take fully advantage of these additional temporal information. Recurrent metworks are networks with loops in them, allowing information to persist [Graves *et al.*, 2013]. A recurrent neuron may look like Fig 5 and when it is unrolled we can see that its output may feed back to itself in future time steps.

LSTM

LSTM is a very successful and popular kind of recurrent neural networks as they are capable of learning long-term dependencies.They were introduced by [Hochreiter and Schmidhuber, 1997], and were refined and popularized by many people. They work tremendously well on a large variety of problems, and are now widely used.LSTMs are explicitly designed to avoid the long-term dependency problem.All recurrent neural networks have the form of a chain of repeating modules of neural network.

In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer.LSTMs also have this chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way.

In Fig 6, each line carries an entire vector, from the output of one node to the inputs of others. The pink circles represent pointwise operations, like vector addition, while the yellow boxes are learned neural network layers. Lines merging denote concatenation, while a line forking denote its content being copied and the copies going to different locations.

In our classification problem, we used LSTM layers to learn a temporal sequence of heroes picking decisions. We use tensorflow [Abadi *et al.*, 2016] as our platform to implement all the neural network methods. Specificaly, our network architecture for this win rate prediction problem is shown in Fig 7. The input of the network is one dimension vectors of length 10 which is described in Section 3.1. The first layer is an embedding layer. This embedding layer is commonly used in Natural Language Processing to addressing the problem to transform index encoding to vector encoding and it is trainable. We do not use one-hot encoding for LSTMs because it will result in very high dimensions and sparse encoding which increase memory footprint and computation time. Specifically, we embed the indexes to 64-dimensional vectors in this experiment.

The second layer is a LSTM layer containing 64 LSTM cells with the structure similar to that in Fig 6. The activation function is $tanh$ for this layer. The LSTM layer is responsible for learning the deep dependencies of the decisions for hero choices. Then the output of the LSTM is activated by a ReLu function and then batch normalized before feeding to the next layer.

The third layer of the network is a densely connected layer which contains 128 neurons. These neurons try to learn the deep latent features extracted by the LSTMs and map them to different classes. The activation function is ReLu for this layer and the output is batch-normalized before feed forward.

The output layer contains two neurons representing Radiant win and Dire Win respectively. The final output is processed by softmax function to conclude a win rate prediction.

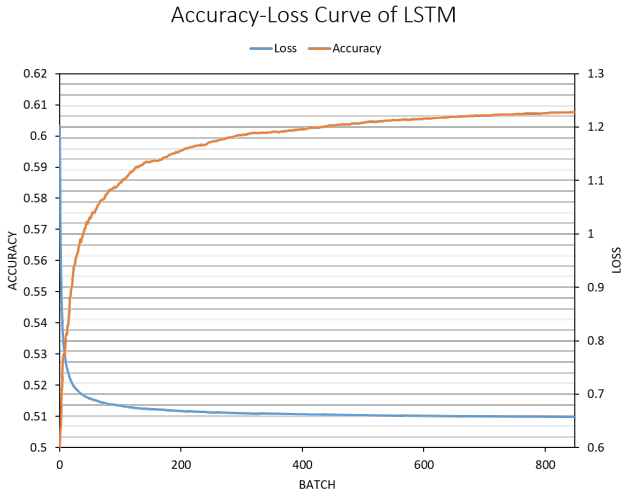

Figure 8: Loss and Accuracy on train data set at each batch

We evaluate our LSTM networks on the cleared 745201 data set. The entire data set is split randomly into a proportion of 0.8 and 0.2 for training data set and testing data set respectively. We use Adam Optimizer [Kingma and Ba, 2015] for this model and the learning rate is 0.01. We trained the model for 2 epochs and the loss curve (sparse categorical crossentropy) is shown in Fig 8. This LSTM combined with DNN

network ultimately got an accuracy of 0.613 at the test data set after the two training epochs, which is not significantly higher than expected considering the usage of a iterative and robust data processing method and a much larger training set. Still, this model gets an advantage of 1-3 percentage in accuracy and is comprehensively evaluated on large enough data set to make sure the result is valid. Without excluding any certain group of data (like using only professional match data), we consider this model still a good candidate.

To evaluate the classification performance more comprehensively we also compute the ROC curve and AUC score of the LSTM networks and it is shown in Fig 9. And the AUC score of this model is 0.651.

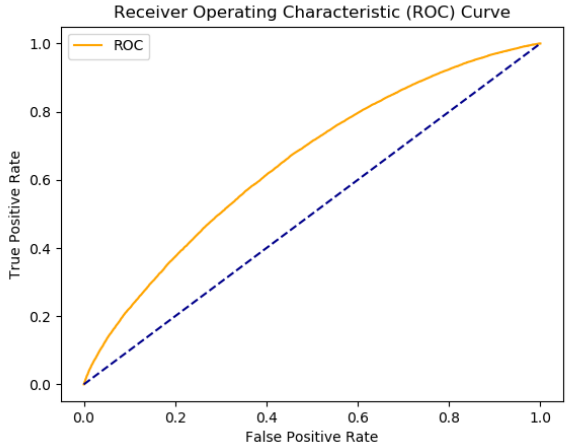

Figure 9: ROC curve of LSTM model, the AUC score is 0.651

4 Discussion

In this section we will discuss the limitations of our prediction model, the limitations of our data processing methods and future works can be done to this problem. Due to the not very satisfying results of the prediction model, we believe there is still much research work can be done about forecasting the result of a Dota2 game. But we still believe that the key problem need to be addressed is how to learn or calculate the true winning rate of a draft. If we cannot label the data correctly, we cannot get significant accuracy gains. If we cannot know the true winning probability, supervised regression is difficult to apply.

We first discuss the limitations of the prediction model. with more complex model and more trainable parameters, the probability of a model to overfit on training data set is higher. We evaluate our prediction model after finish every training epoch and record loss score and accuracy on test data set. While loss keep decreasing and accuracy keep increasing on training set, they turn to drop after just two epochs on testing data set. The LSTMs are still hard to tune and train with being very easy to overfit on training sets even for not very large number of trainalbe parameters. It will take more effort to find optimal parameters and hyper parameters for complex networks consists of various kinds of neural networks. And complex neural networks consumes more computing resources and need much more time to converge. Although this is not often considered a very important limitation, but when the training set is very big or changes a lot , for example Dota2 updates very frequently so that the model needs to be re-trained. I believe further improvement or novel methods need to be proposed to address the problem of training models on highly dynamic data sets.

One of the limitations of the data processing method is that it need to iterate on the complete data set for several times and this consumes a lot of resources and time. And this limitation becomes severe given that Dota2 updates frequently so that this processing may need to be done frequently as well. Despite of the consumption of time and resources, another limitation is that we still cannot explain how well we cleared the data set or how many anomalies are excluded. The quality of processed data is difficult to measure. And essentially we still cannot get to know the true winning probability of certain draft. We believe that to address this classification problem of Dota2 Draft, we need to think up a method to estimate of calculate the true winning probability using a bounded amount of data. This problem is still open and we believe it can contribute to all further problems involving huge data sets of observations without knowing probabilities.

With predicting the outcomes being one of the problems that are not very well addressed, there are still a lot of open problems about Dota2. When OpenAI is trying to test their deep reinforcement algorithms on Dota2, no algorithms are designed to address the draft strategies. Specifically, to built a deep reinforcement algorithm that gives a sequence of hero choices which yields highest win rate against opponents. This algorithm will give professional Dota2 clubs huge benefits as they can use the AI to give advises or train their coaches against it. However, to evaluate whether the AI's choices are good or how good it is, we need to predict the outcomes using the draft. And the prediction accuracy by now is still not very satisfying.

5 Conclusion

In this paper, we showed the importance of predicting the outcome of Dota2 games by analyzing the draft alone and addressed the problem that traditional prediction methods failed to discover deep latent relationships among heroes. We proposed a new machine learning model using LSTM layers combined with DNN layers to learn temporal dependencies among hero choices as well as latent relationships in drafts. And we processed data set using AAE networks based algorithms to robustly remove unrepresentative or wrongly labeled data. As a result, the accuracy and AUC score of our model is 61.3% and 0.651 respectively. Although the accuracy is still not very satisfying, but we showed an effective approach to predict outcomes from flawed data sets when we can not use observations to estimate probabilities.

Appendix

We describe the source code and data sets that are included together with this paper.

1. **aae.py :** implemented the AAE networks used for data processing and the methods we described in section 3.2. The implementation is based on Tensorflow.
2. **lstm.py :** implemented the classification networks including LSTM layers and DNN layers which are described in section 3.3. The implementation is based on Tensorflow.
3. **706e_dataset :** the complete data set used in this paper, which is described in section 3.1.
4. **data_filtered_i.csv :** processed data sets at each iteration using methods described in section 3.2 .

References

[Abadi *et al.*, 2016] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek Gordon Murray, Benoit Steiner, Paul A. Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zhang. Tensorflow: A system for large-scale machine learning. In *OSDI*, 2016.

[Agarwala, 2014] Atish Agarwala. Learning dota 2 team compositions. 2014.

[Beggel *et al.*, 2019] Laura Beggel, Michael Pfeiffer, and Bernd Bischl. Robust anomaly detection in images using adversarial autoencoders. *CoRR*, abs/1901.06355, 2019.

[Conley and Perry, 2013] Kevin E Conley and Daniel Perry. How does he saw me ? a recommendation engine for picking heroes in dota 2. 2013.

[Drachen *et al.*, 2014] Anders Drachen, Matthew Yancey, John Maguire, Derrek Chu, Iris Yuhui Wang, Tobias Mahlmann, Matthias Schubert, and Diego Klabjan. Skill-based differences in spatio-temporal team behaviour in defence of the ancients 2 (dota 2). *2014 IEEE Games Media Entertainment*, pages 1–8, 2014.

[Eggert *et al.*, 2015] Christoph Eggert, Marc Herrlich, Jan Smeddinck, and Rainer Malaka. Classification of player roles in the team-based multi-player game dota 2. In Konstantinos Chorianopoulos, Monica Divitini, Jannicke Baalsrud Hauge, Letizia Jaccheri, and Rainer Malaka, editors, *Entertainment Computing - ICEC 2015*, pages 112–125, Cham, 2015. Springer International Publishing.

[Goodfellow *et al.*, 2014] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

[Graves *et al.*, 2013] Alex Graves, Abdel rahman Mohamed, and Geoffrey E. Hinton. Speech recognition with deep recurrent neural networks. *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649, 2013.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997.

[Johansson and Wikström, 2015] Filip Johansson and Jesper Wikström. Result prediction by mining replays in dota 2. 2015.

[Kalyanaraman, 2015] Kaushik Kalyanaraman. To win or not to win ? a prediction model to determine the outcome of a dota 2 match. 2015.

[Kingma and Ba, 2015] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[Kingma and Welling, 2014] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2014.

[Kinkade, 2015] Nicholas Kinkade. Dota 2 win prediction. 2015.

[Leveau and Joly, 2017] Valentin Leveau and Alexis Joly. Adversarial autoencoders for novelty detection. 2017.

[Makhzani *et al.*, 2015] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, and Ian J. Goodfellow. Adversarial autoencoders. *CoRR*, abs/1511.05644, 2015.

[Nuangjumnonga and Mitomo, 2012] Tinnawat Nuangjumnonga and Hitoshi Mitomo. Leadership development through online gaming. 2012.

[Pobiedina *et al.*, 2013] Nataliia Pobiedina, Julia Neidhardt, María del Carmen Calatrava Moreno, László Grad-Gyenge, and Hannes Werthner. On successful team formation: Statistical analysis of a multiplayer online game. *2013 IEEE 15th Conference on Business Informatics*, pages 55–62, 2013.

[Rioult *et al.*, 2014] François Rioult, Jean-Philippe Métivier, B. Helleu, N. Scelles, and Christophe Durand. Mining tracks of competitive video games. 2014.

[Song *et al.*, 2015] Kuangyan Song, Tianyi Zhang, and Chao Ma. Predicting the winning side of dota 2. 2015.