Yutong He and Yuanzhen Du
CSC 240/440 Data Mining
December 15, 2016
Course Project

Predicting Game Result in Dota 2 by Chat Log

Introduction:

Communication, endlessly contributing to the goal of information exchanging, seems to have been evolved into a more rapid and broader status. Many have noticed the importance of communication while more failed to realize what communication can help them to achieve. This case, interestingly, can be applied to one of the most popular games in the world, Dota 2.

Dota 2 is a free-to-play multiplayer online battle arena (MOBA) game. Match is between two teams that each consists of five players, with both teams occupying their own separate base on the map. Each of the ten players independently control a powerful character –"hero" and perform unique abilities and different styles of playing. Players collect experience points and items for heroes in order to push through the opposing team's defenses. A team wins by being the first to destroy the opposing team's base -"Ancient".

[8]

Dota 2 is well known for the annual tournament hosted by Valve. The sixth international just ended this summer with a total prize pool of 20 million dollars. With the properties of high competitive and heavily relying on teamwork. It now has average 600 thousands players with peak players at 1.2 million [7]. It is also well balanced and has highly demand on skills. With more and more median coverage and exposure including live streaming and its competition. It has been a competitive sport with huge potential of economical profits. There is a trend that more professional teams will hire data analysts to analyze team's practice and match performance data.

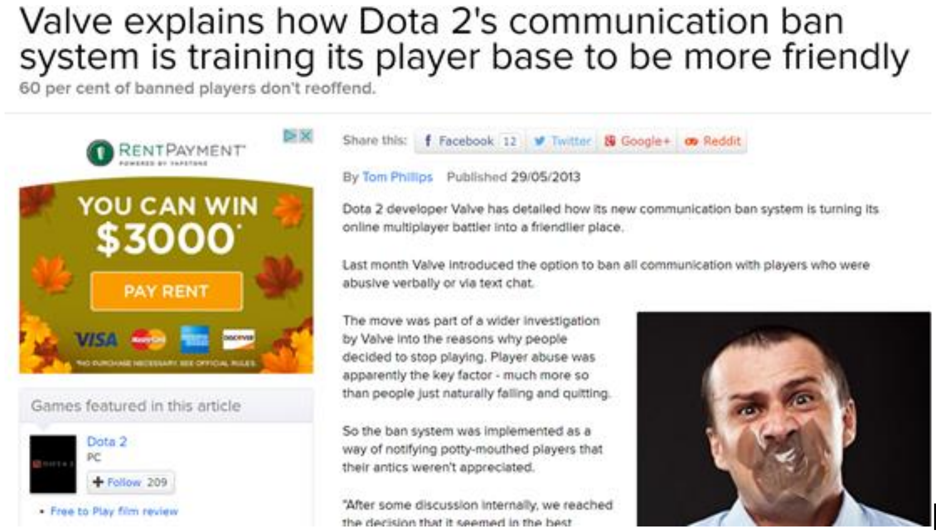
Valve releases 59 pages stats to teams after alleged leak

Posted by Andrew "Briticus" Henderson at 25 July 2015 03:25

[9]

From professional on T.I to normal matches on steam, all players need good communications for team fight to achieve goals. We decide to investigate some interesting facts about chatting and we get 50 thousands match records which include players' chat contents and other gaming stats such as experience and gold gains. Our goal is to mine through the data and mainly focus on extracting useful information about chats in game.

Even though play skills is huge factor affecting the game, the communication relating to information exchange such as attacking order and strategy plans is still pretty important to the result of the game. Due to relatively anonymous "environment", people tend to express freely and may type in rude language like taunting and teasing opponents or teammates. Chats reflect the fighting process somehow and it also affects players' mood and actions. In addition, Valve seems to have the need of developing a strategy to detect those players who are frequently unfriendly to others during the game, in that situation our research will be available for them.



Valve explains how Dota 2's communication ban system is training its player base to be more friendly

60 per cent of banned players don't reoffend.

By Tom Phillips Published 29/05/2013

Dota 2 developer Valve has detailed how its new communication ban system is turning its online multiplayer battler into a friendlier place.

Last month Valve introduced the option to ban all communication with players who were abusive verbally or via text chat.

The move was part of a wider investigation by Valve into the reasons why people decided to stop playing. Player abuse was apparently the key factor - much more so than people just naturally falling and quitting.

So the ban system was implemented as a way of notifying potty-mouthed players that their antics weren't appreciated.

"After some discussion internally, we reached the decision that it seemed in the best

[10]

Literature Review:

Previous work related to Dota 2 seems to focus on two aspects: the players' roles in successful team building [1] [5] which pay more attention on the player's characters and their effects on the team work and the result of the game, and the hero picking[2] and skill -based team behavior[3] which mainly discuss about the particular in-game strategies for winning the game.

Among the researches on Dota 2, Pobiedina brings up a novel idea about using game log data and statistical approaches to identify the general factors affecting the result of the game [4]. In her paper, she concludes four factors which can be considered as the most important aspects of the success of a team: (1) a good role distribution; (2) experienced players; (3) playing with friends; (4) the selection of a proper leader.

While the first two aspects have been widely discussed as we mentioned at the beginning of the literature review, the last two factors from Pobiedina's paper inspire us more. These remind us that Dota 2 is a team-based multiplayer online game and social factors do make a difference in terms of winning or losing the game. As a result, we are curious about how in-game communication, which can be considered as one of the most socialized aspects of Dota 2, takes part in determining the result of the game.

Problem Statement:

Our purpose for this project is to classify and predict whether a team will win or lose in a Dota 2 game based on their in-game chat log.

Unlike the previous researches on predicting the game result in Dota 2, our prediction does not consider experience gain, gold gain, KDA (kills, deaths and assistance) information and other game playing data which are commonly used in other researches. We only consider the in-game chat contexts and their occurrence time as features of our classification.

Data Source:

We acquire the dataset of 50000 Dota 2 ranked ladder matches on Kaggle (Link: https://www.kaggle.com/devinanzelmo/dota-2-matches). It contains (1) the table of chat log, which includes attributes of match ID, key words, occurrence time, and user ID, (2)the table of matches information, which includes the attributes of time, towers status, barracks status, the occurrence time of important events such as first blood and the result of the game, (3) the table of players, which includes the attributes of experience gain, gold gain, and KDA (kills, deaths and assistance) information, (4) the tables of team fight information, players' performance in team fight, hero names, and objectives information.

Methods:

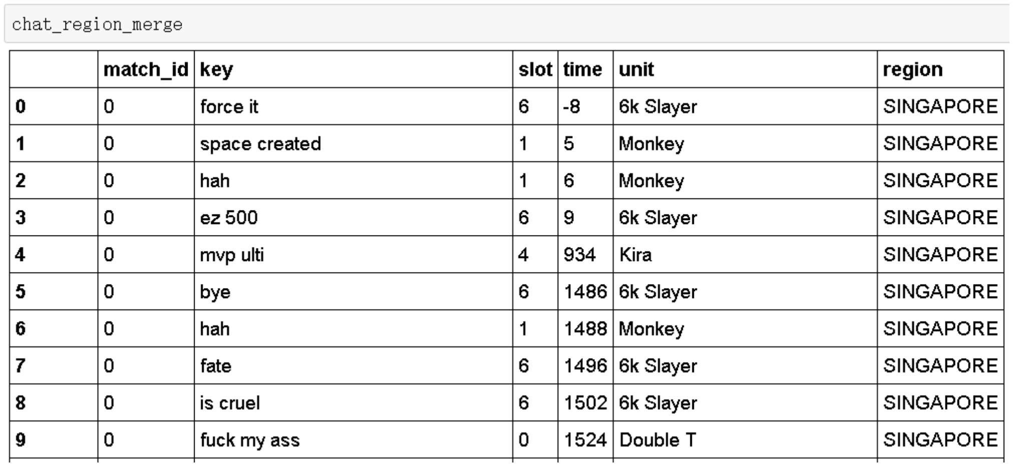
We construct our project by the following steps. Each will be discussed in separate subsections.

1. Data Preprocessing
2. Basic Data Analysis
3. Natural Language Processing and Word Cloud Generation
4. Feature Engineering
    a. Bag of Words
    b. Tf-idf
    c. Sentiment Analysis
    d. Analysis by Time Interval
5. Classification
    a. Support Vector Machine
    b. Naive Bayes
    c. Random Forest

I. Data Preprocessing

The key attributes spread in raw data sets' different parts which include different information. We used Python Panda to do preprocess by merging and filtering to generate clean, neat data frame.

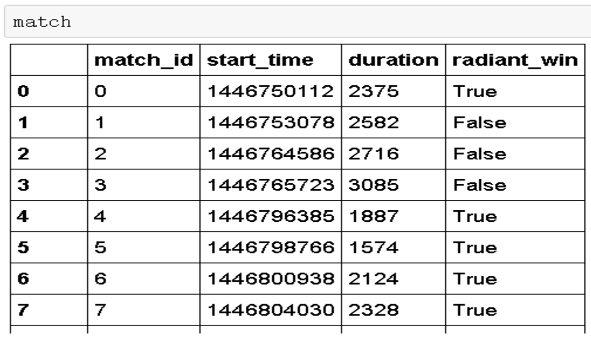
- Chat and Cluster merge to filter undesired area.



```
chat_region_merge
```

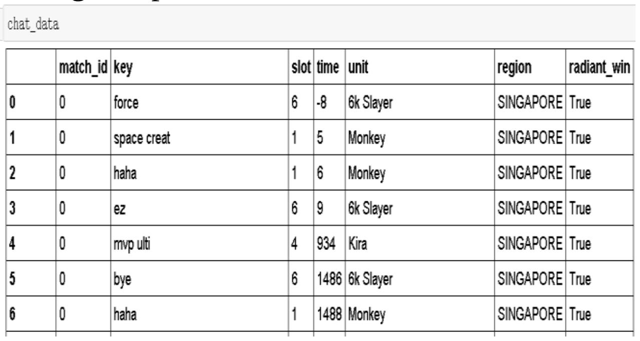
	match_id	key	slot	time	unit	region
0	0	force it	6	-8	6k Slayer	SINGAPORE
1	0	space created	1	5	Monkey	SINGAPORE
2	0	hah	1	6	Monkey	SINGAPORE
3	0	ez 500	6	9	6k Slayer	SINGAPORE
4	0	mvp ulti	4	934	Kira	SINGAPORE
5	0	bye	6	1486	6k Slayer	SINGAPORE
6	0	hah	1	1488	Monkey	SINGAPORE
7	0	fate	6	1496	6k Slayer	SINGAPORE
8	0	is cruel	6	1502	6k Slayer	SINGAPORE
9	0	fuck my ass	0	1524	Double T	SINGAPORE

- Filtered Match_Results merged with Chat to get important frame with win/lost details.



```
match
```

	match_id	start_time	duration	radiant_win
0	0	1446750112	2375	True
1	1	1446753078	2582	False
2	2	1446764586	2716	False
3	3	1446765723	3085	False
4	4	1446796385	1887	True
5	5	1446798766	1574	True
6	6	1446800938	2124	True
7	7	1446804030	2328	True

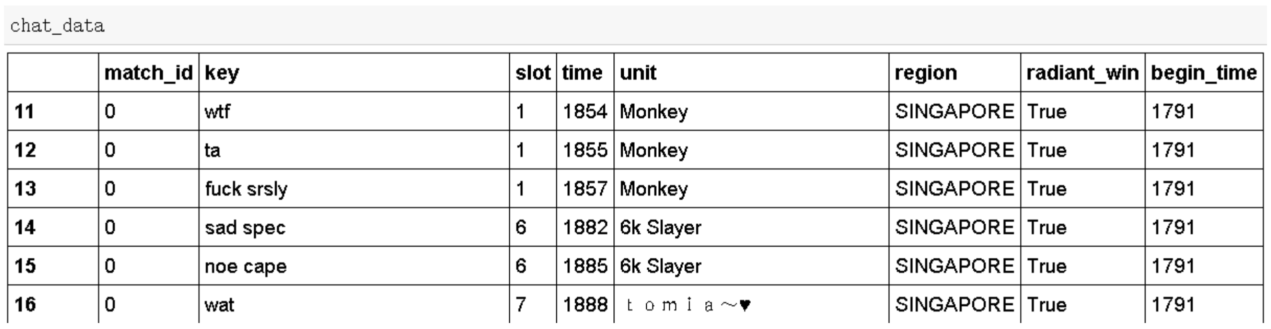


```
chat_data
```

	match_id	key	slot	time	unit	region	radiant_win
0	0	force	6	-8	6k Slayer	SINGAPORE	True
1	0	space creat	1	5	Monkey	SINGAPORE	True
2	0	haha	1	6	Monkey	SINGAPORE	True
3	0	ez	6	9	6k Slayer	SINGAPORE	True
4	0	mvp ulti	4	934	Kira	SINGAPORE	True
5	0	bye	6	1486	6k Slayer	SINGAPORE	True
6	0	haha	1	1488	Monkey	SINGAPORE	True

- Win and Lost separated data frame for future Win/Lost analysis.

- Team_fights with Chat_data combined to pick chat in certain interval. Time interval may relate to Win/Lost chatting pattern. Chat happening time later than begin_time will be choose.



chat_data

	match_id	key	slot	time	unit	region	radiant_win	begin_time
11	0	wtf	1	1854	Monkey	SINGAPORE	True	1791
12	0	ta	1	1855	Monkey	SINGAPORE	True	1791
13	0	fuck srsly	1	1857	Monkey	SINGAPORE	True	1791
14	0	sad spec	6	1882	6k Slayer	SINGAPORE	True	1791
15	0	noe cape	6	1885	6k Slayer	SINGAPORE	True	1791
16	0	wat	7	1888	ｔｏｍｉａ～♥	SINGAPORE	True	1791

II. Basic data analysis

For statistical analysis, we focus on game regions where main language is English likewise Singapore, U.S, Australia and Europe. Several analysis generated: 1. Chat lines counts on match level (average for each match) and on player level (average per person). 2. Distribution analysis by histogram and its log scale presented. 3. Win and Lost sides analysis, basically similar statistical analysis as before to see if there are significant different chatting patterns between win and lost team.

III. Natural Language Processing and Word Cloud generation

A. Natural Language Processing

After we obtained the proper chat data, we used strategies in natural language processing to acquire a better qualification of our data.

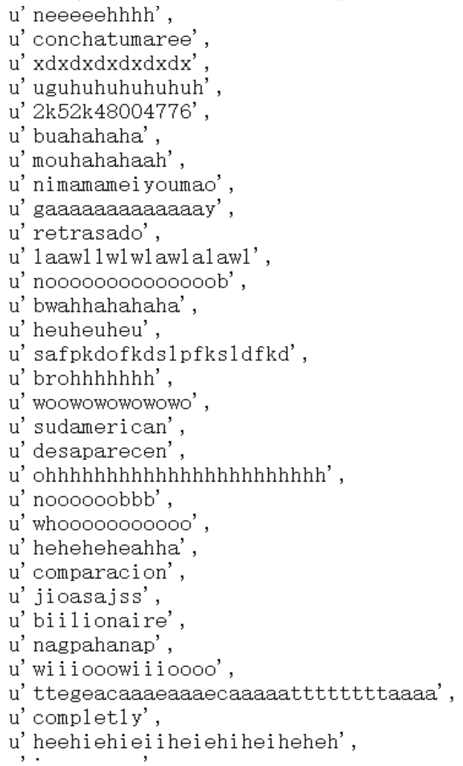
We first separated all the winning team chat log and losing team chat log into two different documents. Next, removed the non-English words by defining a function that takes a string as it parameter and returns a filtered string without non-ascii words. Then we tokenized the returned strings and conditionally stemmed them by using SnowballStemmer[11] from NLTK[12] package. Because the SnowballStemmer deletes the last "e" in the "e"-ending words and replaces the last "y" into "i" in the "y"-ending words, so we skipped the stemming when encountered "e"-ending words and partially converted back the "y"-ending words if they are not one of the commonly occurring "i-ending" words such as "hi" and "ulti".

Punctuations, words of digit type only words are ignored because they don't provide much useful information. We then check if a word's non duplicated characters (words like 'haaahaaa', 'fckkkkkkk', 'ggwp' and 'wppp') is a subset of ('h','a'), ('f','u','c','k') or ('g','w','p') and they are all converted to 'haha', 'fuck' and 'gg'.

Common typos and synonyms are corrected by using simple conditionals and used a mini library that we wrote to replace frequently occurring stopword abbreviations used in Dota 2 to their original words. For example, we converted “u” into “you” and “ur” into “your”. Then we combined our mini library and the stopword library from the nltk.corpus[13] to eliminate the stopwords.

Especially for words like 'im', 'u', 'ur', 'ru', 'youre', 'hes', 'shes', 'its' after we remove the punctuations, those actually can’t be detected by any stemmer in nltk, so that we have to screen them out manually. These words turned out to be the common stopwords however players tend to express it differently.

When we peeked the frequency distribution results from tokenizer of those words, we found out that there are many contents full of duplicated characters.



```
u'neeeeehhhh',
u'conchatumaree',
u'xdxdxdxdxdxdx',
u'uguhuhuhuhuhuh',
u'2k52k48004776',
u'buahahaha',
u'mouhahahaah',
u'nimamameiyoumao',
u'gaaaaaaaaaaaaay',
u'retrasado',
u'laawllwlwlawlalawl',
u'noooooooooooooob',
u'bwahhahahaha',
u'heuheuheu',
u'safpkdofkdslpfksldfkd',
u'brohhhhhhh',
u'woowowowowowo',
u'sudamerican',
u'desaparecen',
u'ohhhhhhhhhhhhhhhhhhhhhhhh',
u'noooooobbb',
u'whoooooooooo',
u'heheheheahha',
u'comparacion',
u'jioasajss',
u'biilionaire',
u'nagpahanap',
u'wiiiooowiiioooo',
u'ttegeacaaaeaaaecaaaaatttttttttaaaa',
u'completly',
u'heehiehieiiheiehiheiheheh',
```

Words length longer than 8 character are shown above and we think it is safe to remove duplicates by set() command in Python. Even Though there are meaningful words in the list, it turns out it still maintain full information after we remove its duplicates. Additionally after processing 10097 of them, we did get a huge boost of enlarging the precise words group.

After we deleted those empty chat tuple, finally we get a pretty precise and neat chat data set. Large portion of nonsense junk are eliminated and players’ varies ways of expressions reformed to a general meaningful form. This is probably the most important part for this research because this clean word text are heavily used for analysis, feature extractions, classification and predictions. It not only increase the processing speed and improve the accuracy.

B. Word Cloud Generation

After we cleaned our data by data preprocessing and natural language processing, it is important for us to visualize our data so that we can determine the strategies and algorithms in feature engineering and the final classification. As a result, we created word clouds for winning teams, losing teams and the whole data set.

We used the WordCloud[14] from wordcloud package to generate the word clouds that contain the top 100 most frequent words in all of the three groups.

IV. Feature Engineering

A. Bag of words

The first thing we did to generate the features for classification is to create a bag of words model. It takes the chat log that has gone through the natural language processing process of each team in each game as a document and count the occurrence of each word that appears in the document to generate the features we need for training our classifier.

We used the CountVectorizer[15] from the feature extraction of the sklearn package (sklearn.feature_extraction) to count and generate the features needed, which are the frequencies of the words that commonly occur. We have tried several different numbers of features on different models which will be discussed on the following sections. The features generated here that we mostly used are the frequencies of the top 500 words that commonly occur.

After bagging the words, we are able get a numeric matrix that represents our original text data. In the matrix, each row represents a data tuple - a team in one game, and each column represents a feature - a word under which are its occurrences in team chats.

B. Tf-idf

After we obtained the bag of words model, we applied tf-idf, term frequency - inverse document frequency to better represent the information in the data. TF-idf uses the product of two statistics, the term frequency and the inverse of the document frequency to represent the importance of a word to a document and its class label. A word can have high tf-idf value if it occurs frequently in the document but is not frequently used in general.

We used the TfidfTransformer[16] in sklearn.feature_extraction.text package to acquire the tf-idf value matrix as our new feature matrix that potentially increases the accuracy of classification.

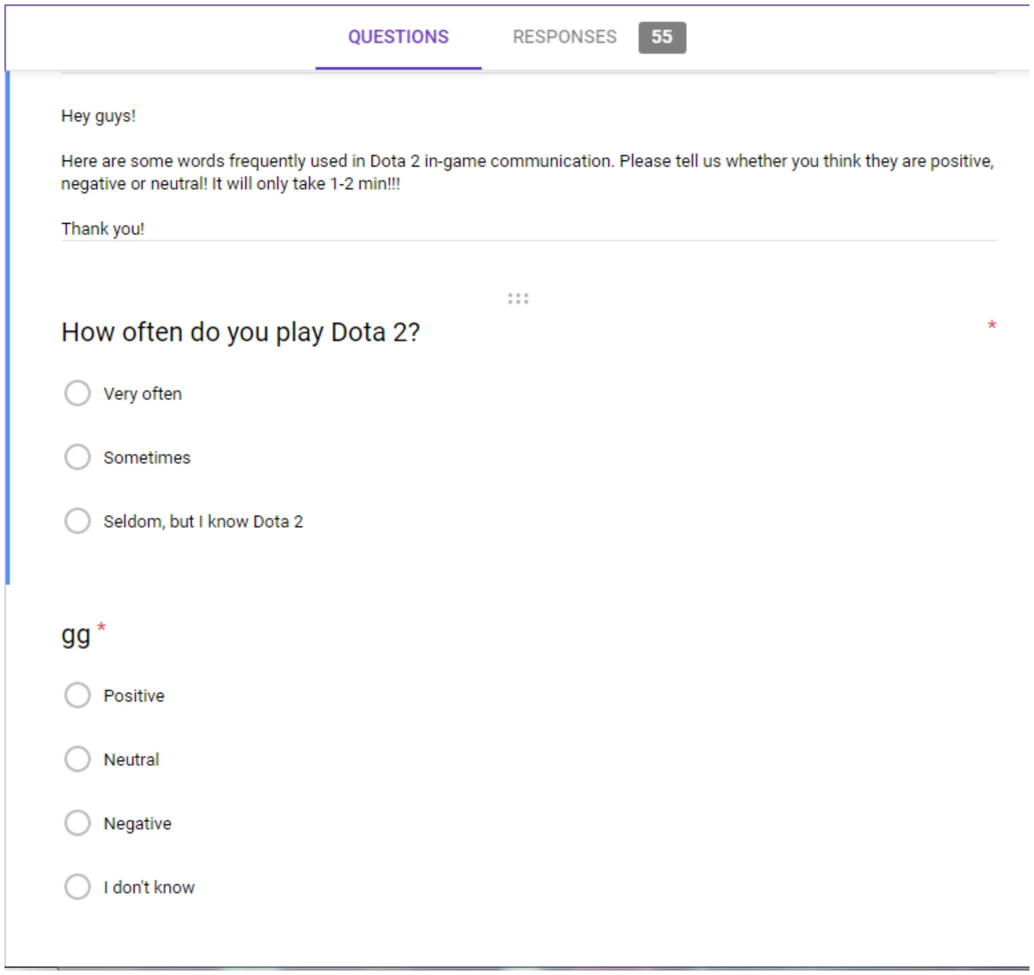
In addition to perform tf-idf to the feature matrix obtained from original feature matrix, which considers each team chat log as an independent document, we also all winning and losing chat as the only two documents and perform tf-idf on this matrix with only two rows. In this matrix, if the tf-idf values of a certain feature is lower than a threshold in both row, we removed that feature in the original feature matrix and by iterating through every feature we picked, we generated the third feature matrix.

Because we reduced the dimension in the third feature matrix, it potentially reduce the training and testing time.

C. Sentiment analysis

As a very important part of natural language processing and text mining, sentiment analysis is an aspect of our data that we definitely do not want to ignore. However, there are a lot of terminologies and abbreviations that basically only appear in Dota 2 and nowhere else and players usually do not type complete and grammatically correct sentences. These cause the traditional ways of sentimental test becoming relatively unreliable.

To make the result of our sentiment analysis more accurate, we selected 36 commonly used proper nouns and abbreviations and created a survey for Dota 2 players to evaluate whether the words are positive, negative or neutral.



Hey guys!

Here are some words frequently used in Dota 2 in-game communication. Please tell us whether you think they are positive, negative or neutral! It will only take 1-2 min!!!

Thank you!

How often do you play Dota 2? *

- Very often
- Sometimes
- Seldom, but I know Dota 2

gg *

- Positive
- Neutral
- Negative
- I don't know

We send out our survey through social Medias such as Facebook and Wechat. Each person who takes the survey assigns a sentimental value to the words. We defined 1 as extremely positive, -1 as extremely negative and 0 as extremely neutral. After we collected the evaluations from players, we weighted their responses differently by taking account of how often do they play the game. The more often the player plays the game, the higher his/her response will be weighted. After we collected all the responses, we ignored the “I don’t know” data points and calculated the compound sentimental scores of the words by taking the mean of their evaluated values and normalized them to the range of -1 to 1. In this way, we generated a unique dictionary which maps the 36 words to their newly evaluated sentimental scores.

For the words that are not in the dictionary, we used the Sentiment Intensity Analyzer [17] from the vaderSentiment package which uses the similar scoring system as ours to evaluate their sentimental score.

By combining the two methods above, we defined our own sentiment analysis method specifically for Dota 2 in-game communication which returns the word-wise sentimental scores. By averaging the scores in each document, we are able to get the sentimental score of the document.

D. Analysis by Time Interval

For Dota 2 matches, in different time intervals, game processes are significant different. In that situation we assume players chatting differently in different time intervals, and it is reasonable to individually analysis chat content for different intervals. For example First Blood (When first hero get killed in the game) or Team Fights intervals for people tend to chat a lot after each fights. Particularly, the last quarter of the game will be a highly related to match results. As concluded, we choose to analyze the chat in the last quarter of games to see if it give us improvements in distinguishing win/lost patterns.

V. Classification

A. Support Vector Machine

The first classification algorithm we tried to use is the support vector machine because we are trying to perform a binary classification. We used the svm[18] from sklearn package to construct our classifier. Currently we have tried two types of kernels - radial basis function kernel and linear kernel. Note that since linear kernel performs relatively poor running time, we only applied it to several but not all of our feature matrices.

B. Naive Bayes

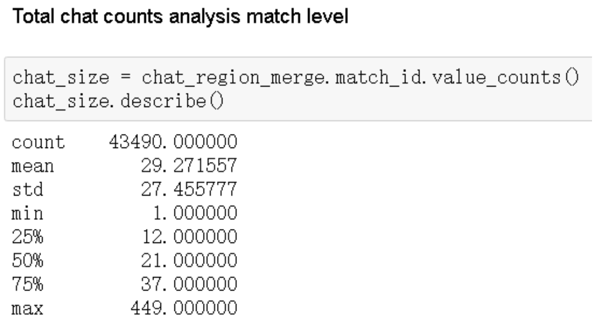
Another classifier we tried is naive bayes. We chose it because of its fast running time and because we removed the stopwords and conjunctions and players usually use single word or short incomplete sentences to express themselves, it is reasonable for us to assume that the occurrence of each word is independent, although this assumption may lower the accuracy of our classification. We used GaussianNB[19] in naive bayes from sklearn package to construct our naive bayes classifier.

C. Random Forest

We also considered random forest as one of the classifieds that worth to be tried because it is intuitive to consider the class label with most vote among the trees as the class label of the team chat. Also, because random forest usually can handle high dimensional data set pretty well and usually relatively high accuracy, we decided to train random forest models in our different feature matrices as well. We used RandomForestClassifier[20] from ensemble of sklearn to construct our random forest classifier.

Result and Analysis:

I. Basic Data Analysis

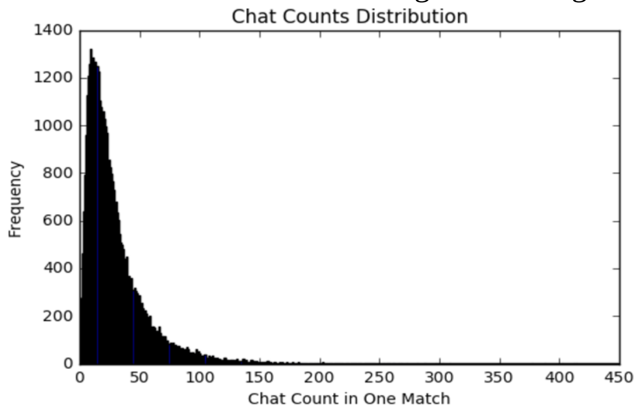


Total chat counts analysis match level

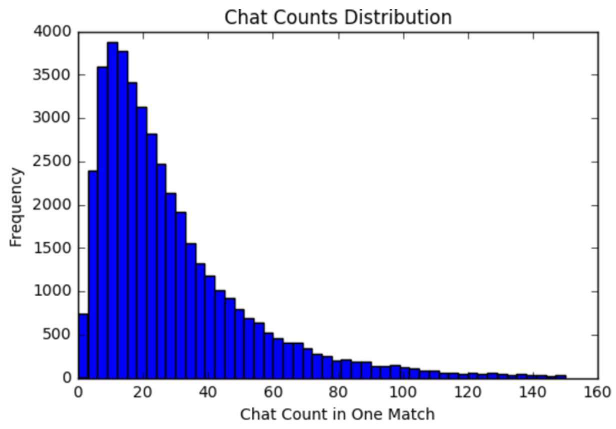
```
chat_size = chat_region_merge.match_id.value_counts()
chat_size.describe()
```

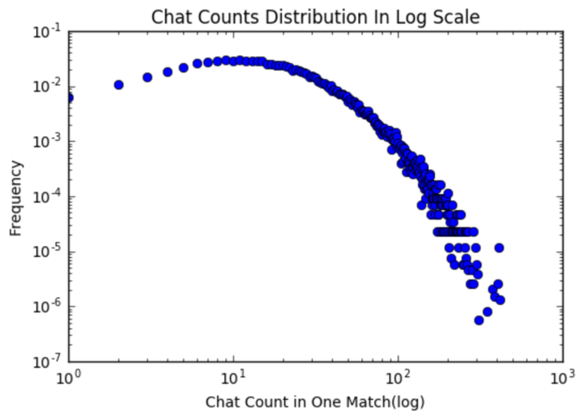
```
count    43490.000000
mean        29.271557
std         27.455777
min          1.000000
25%         12.000000
50%         21.000000
75%         37.000000
max        449.000000
```

Results are simply showed above, with mean of 30, min of 1 and max for 449. We are kind of surprised that some matches with have one chat during the whole game.

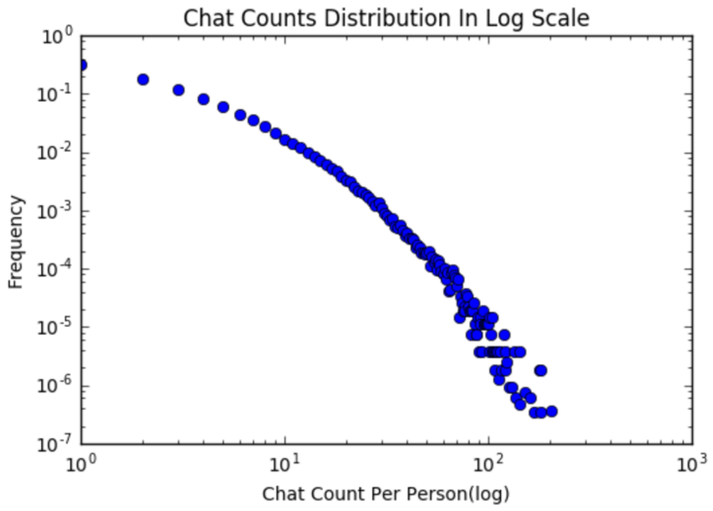


The distribution is heavily skewed to the right, and below we see the detail of the distributions between 0 and 160 counts.

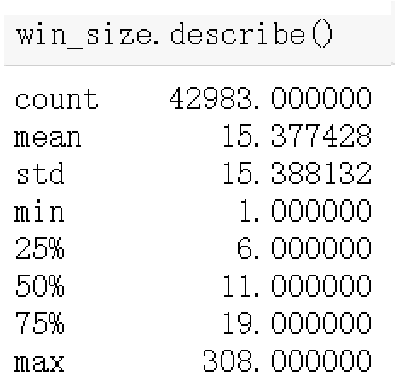




Even after taking a log scale, it is neither power law distributed nor normally distributed.

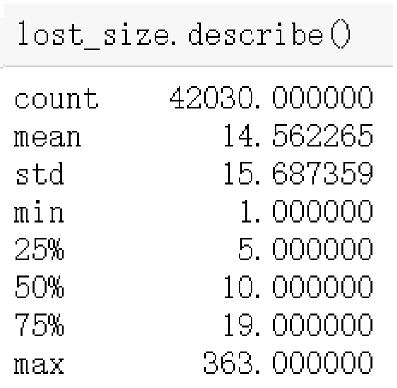


Similar analysis is made for individual level, and for its log form, it can be better fitted by a power law regression.



```
win_size.describe()

count    42983.000000
mean        15.377428
std         15.388132
min          1.000000
25%          6.000000
50%         11.000000
75%         19.000000
max        308.000000
```



```
lost_size.describe()

count    42030.000000
mean        14.562265
std         15.687359
min          1.000000
25%          5.000000
50%         10.000000
75%         19.000000
max        363.000000
```

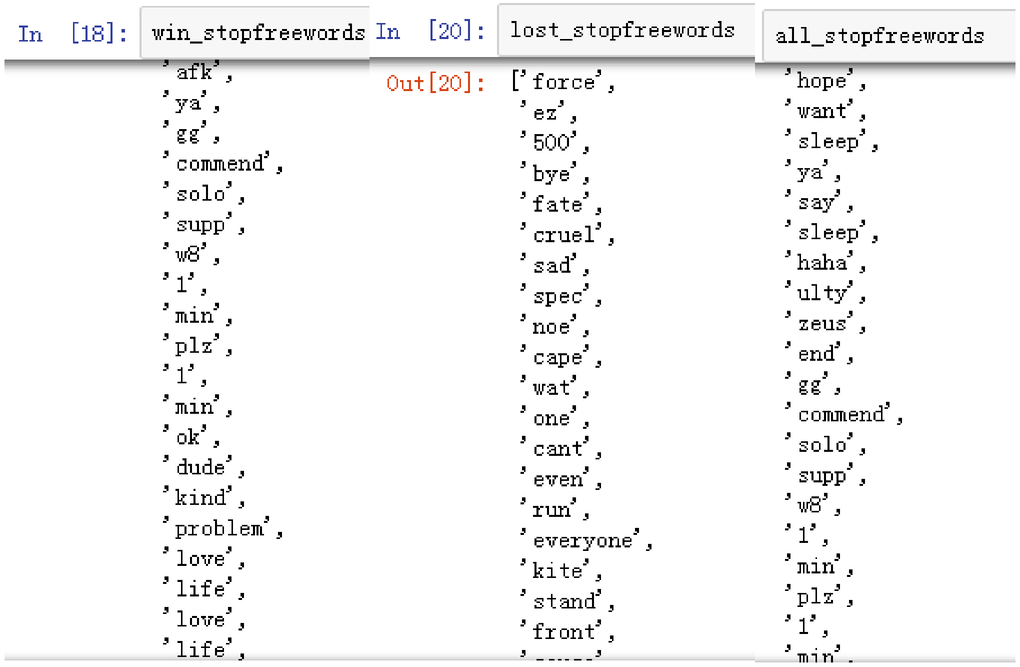
Statistical analysis of chat counts for win/lost sides are really similar. A small piece of information that max counts of lost surpassed nearly ⅙ of the winning team, for that it is probably that lost team tend to debate with others a lot.

fuck
report
ez
gg
lol
haha
xd
game
mid
noob
nice
team
commend
kill
ty

Here are a little visual comparisons of a word cloud when win at left and lost at right. They are indeed similar and we need detail statistical analysis on those words frequency for comparison between two groups. We are going to discuss more about the word cloud in the next section.

II. Natural Language Processing and Word Cloud Generation

After we processed the data cleaning based on the natural language processing strategies we mentioned above, we obtain the new chat log for winning teams, losing teams and all teams.



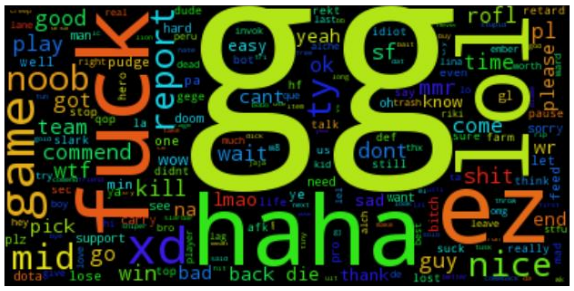
```
In  [18]: win_stopfreewords
 'afk',
 'ya',
 'gg',
 'commend',
 'solo',
 'supp',
 'w8',
 '1',
 'min',
 'plz',
 '1',
 'min',
 'ok',
 'dude',
 'kind',
 'problem',
 'love',
 'life',
 'love',
 'life',
```

```
In  [20]: lost_stopfreewords
Out[20]: ['force',
 'ez',
 '500',
 'bye',
 'fate',
 'cruel',
 'sad',
 'spec',
 'noe',
 'cape',
 'wat',
 'one',
 'cant',
 'even',
 'run',
 'everyone',
 'kite',
 'stand',
 'front',
```

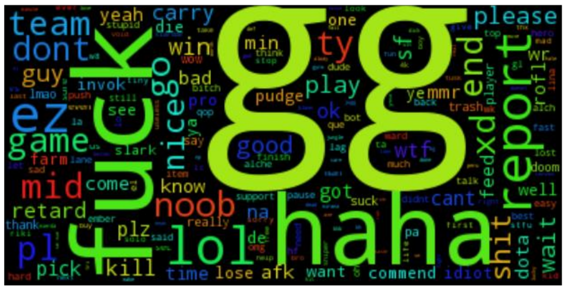
```
all_stopfreewords
 'hope',
 'want',
 'sleep',
 'ya',
 'say',
 'sleep',
 'haha',
 'ulty',
 'zeus',
 'end',
 'gg',
 'commend',
 'solo',
 'supp',
 'w8',
 '1',
 'min',
 'plz',
 '1',
 'min',
```

Then based on the top 100 words in the new chat log, we generated the word cloud for all three of the team groups.

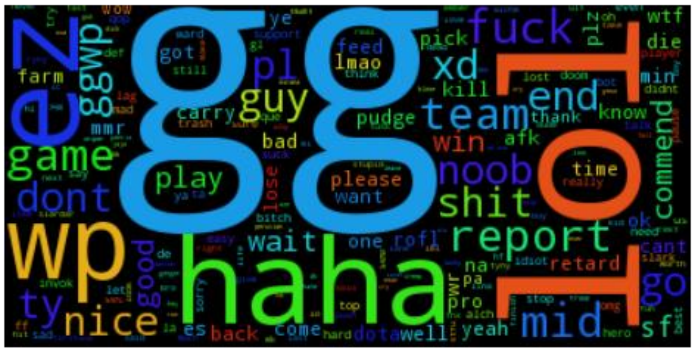
Word Cloud for winning teams



Word Cloud for losing teams



Word Cloud for all teams

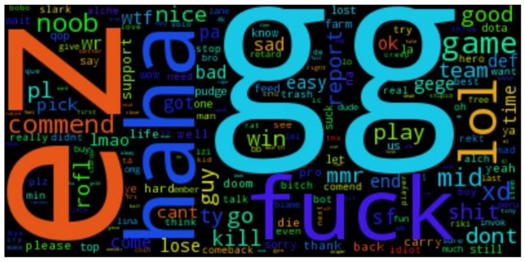


By the word cloud we can see that although there are some words such as “gg” and “haha” frequently occurring in both winning team and losing team, some of the words such as “ez” and “report” have legitimately different frequencies in two groups.

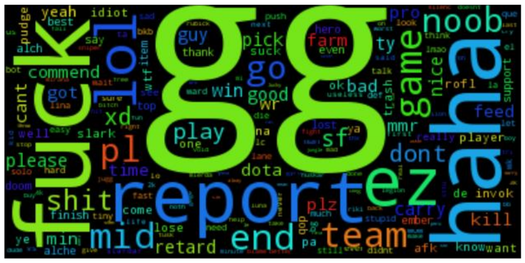
This result established our faith in using bag of word to generate feature matrix for our classification.

Moreover, after we separated out and cleaned the chat data for the chat log in the last ¼ of games, we obtained a pair of more comparable word cloud.

Winning teams in the last ¼ of games



Lost teams in the last ¼ of games:

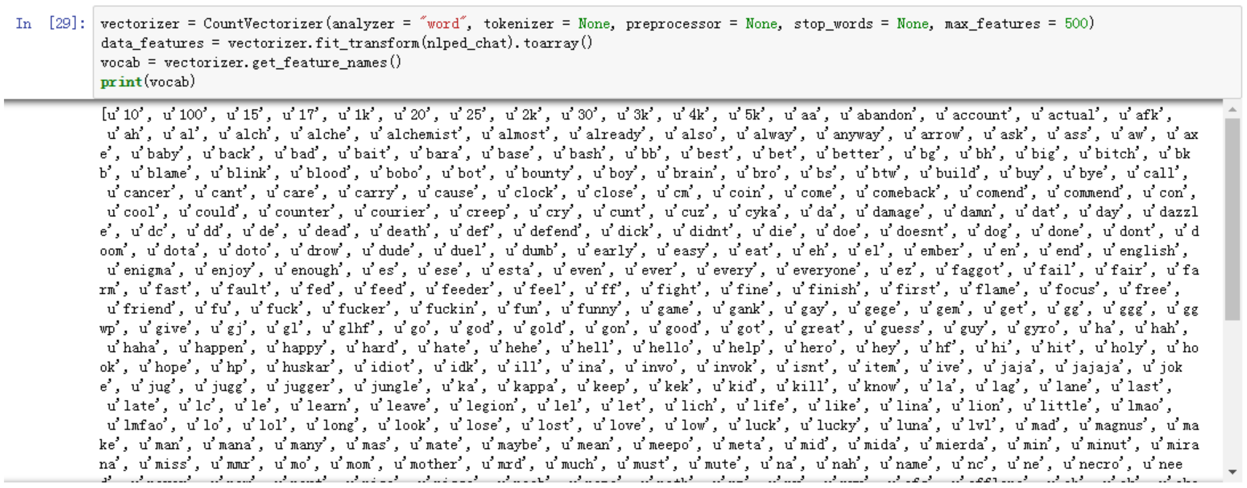


The difference between the frequent words of the two groups becomes more obvious in the last ¼ of games. From here we were convinced that grouping chat data in different time periods and sentimental tests is going to improve our results of classification.

III. Feature Engineering

Even though we were convinced that the sentimental test will improve our classification, after we performed the sentiment analysis on both winning team and losing team separately, the average score of winning team is about +0.02 and the average score of losing team is about + 0.01. Considering there is no much difference between the two scores and the sentimental test function is really time consuming, we decided not to consider the sentimental score as an attribute in our classification.

With bag of words only we generated the feature matrices for both winning team and losing team. Here are the vocabularies that we chose for our most frequently used matrices, the one with 500 features.



```
In  [29]: vectorizer = CountVectorizer(analyzer = "word", tokenizer = None, preprocessor = None, stop_words = None, max_features = 500)
          data_features = vectorizer.fit_transform(nlped_chat).toarray()
          vocab = vectorizer.get_feature_names()
          print(vocab)
```

```
[u'10', u'100', u'15', u'17', u'1k', u'20', u'25', u'2k', u'30', u'3k', u'4k', u'5k', u'aa', u'abandon', u'account', u'actual', u'afk',
 u'ah', u'al', u'alch', u'alche', u'alchemist', u'almost', u'already', u'also', u'alway', u'anyway', u'arrow', u'ask', u'ass', u'aw', u'ax
e', u'baby', u'back', u'bad', u'bait', u'bara', u'base', u'bash', u'bb', u'best', u'bet', u'better', u'bg', u'bh', u'big', u'bitch', u'bk
b', u'blame', u'blink', u'blood', u'bobo', u'bot', u'bounty', u'boy', u'brain', u'bro', u'bs', u'btw', u'build', u'buy', u'bye', u'call',
 u'cancer', u'cant', u'care', u'carry', u'cause', u'clock', u'close', u'cm', u'coin', u'come', u'comeback', u'comend', u'commend', u'con',
 u'cool', u'could', u'counter', u'courier', u'creep', u'cry', u'cunt', u'cuz', u'cyka', u'da', u'damage', u'damn', u'dat', u'day', u'dazzl
e', u'dc', u'dd', u'de', u'dead', u'death', u'def', u'defend', u'dick', u'didnt', u'die', u'doe', u'doesnt', u'dog', u'done', u'dont', u'd
oom', u'dota', u'doto', u'drow', u'dude', u'duel', u'dumb', u'early', u'easy', u'eat', u'eh', u'el', u'ember', u'en', u'end', u'english',
 u'enigma', u'enjoy', u'enough', u'es', u'ese', u'esta', u'even', u'ever', u'every', u'everyone', u'ez', u'faggot', u'fail', u'fair', u'fa
rm', u'fast', u'fault', u'fed', u'feed', u'feeder', u'feel', u'ff', u'fight', u'fine', u'finish', u'first', u'flame', u'focus', u'free',
 u'friend', u'fu', u'fuck', u'fucker', u'fuckin', u'fun', u'funny', u'game', u'gank', u'gay', u'gege', u'gem', u'get', u'gg', u'ggg', u'gg
wp', u'give', u'gj', u'gl', u'glhf', u'go', u'god', u'gold', u'gon', u'good', u'got', u'great', u'guess', u'guy', u'gyro', u'ha', u'hah',
 u'haha', u'happen', u'happy', u'hard', u'hate', u'hehe', u'hell', u'hello', u'help', u'hero', u'hey', u'hf', u'hi', u'hit', u'holy', u'ho
ok', u'hope', u'hp', u'huskar', u'idiot', u'idk', u'ill', u'ina', u'invo', u'invok', u'isnt', u'item', u'ive', u'jaja', u'jajaja', u'jok
e', u'jug', u'jugg', u'jugger', u'jungle', u'ka', u'kappa', u'keep', u'kek', u'kid', u'kill', u'know', u'la', u'lag', u'lane', u'last',
 u'late', u'lc', u'le', u'learn', u'leave', u'legion', u'lel', u'let', u'lich', u'life', u'like', u'lina', u'lion', u'little', u'lmao',
 u'lmfao', u'lo', u'lol', u'long', u'look', u'lose', u'lost', u'love', u'low', u'luck', u'lucky', u'luna', u'lvl', u'mad', u'magnus', u'ma
ke', u'man', u'mana', u'many', u'mas', u'mate', u'maybe', u'mean', u'meepo', u'meta', u'mid', u'mida', u'mierda', u'min', u'minut', u'mira
na', u'miss', u'mmr', u'mo', u'mom', u'mother', u'mrd', u'much', u'must', u'mute', u'na', u'nah', u'name', u'nc', u'ne', u'necro', u'nee
```

For support vector machine and naive bayes, we also tried several different numbers of features such as 30, 35, 50, 100, 275, and 1000. Their effects will be discussed in the following sections.

For each matrices generated, we also performed tf-idf and feature selection which we have introduced in the previous sections on them to generate matrices with tf-idf values and the maintained dimensions, and the matrices with original frequencies and the reduced dimensions. Generally the dimension reduction can cut off half of the original features.

As we discussed above, because people tend to say things more relating to winning and losing trend, so that last ¼ time interval chat are used to do classifications. The detailed comparison and effects of different feature choices will be discussed with the result of classification and its evaluation in the next section.

IV. Classification and Evaluation

	SVM-rbf	**Naive Bayes**	**Random Forest**	**SVM-linear**

Bag of Words	67.555725%	57.086985%	66.523555%	NaN
Tf-idf	65.882491%	60.174675%	66.732341%	66.823502%
Feature Selection	67.382227%	61.653826%	65.867788%	NaN
Last Quarter	67.509850%	59.937390%	67.060072%	68.652285%
Last Quarter tf-idf	66.563818%	62.096216%	67.366664%	67.451671%

The table above is the accuracy evaluation results of different classifiers and feature matrices with 500 features in each.

We used cross-validation with randomly chosen testing data set of size 0.4 of the whole data set to evaluate our classifiers.

The prior threshold is 0.503646415338, calculated by taking the average of the class labels (1 for win and 0 for loss) of the data tuples in the testing data set.

For naive bayes, we have tried the number of features of 30, 35, 50, 275 and got the best result at 35, which performed equally in all cases with accuracy of around 61%.

For support vector machine, we have tried the number of features of 275, 300 and 1000 and got similar result with those numbers.

All in all, SVM, Random Forest and SVM-linear have pretty good results which is all above 65% accuracy, while Naive Bayes processed really fast with relative low accuracy nearly in all situations except SVM-linear classifier. SVM-learn's results don't show much difference with SVM when processed much faster anyway. In the end, the choice of analysis last quarter time period data indeed provide us a pretty good results with impressing speed improvement.

Conclusion:

Even though from most people's intuition chat seems to be an unrelated part with game's win or lost, we successfully digging out some interesting patterns basing it. The chatting text process plays an important role to help us reducing enumerate of noise and making patterns more outstanding and clear. That is really convenient for the future analysis. We investigated Dota 2 from a different angle to see how people communicate and that led us to extract patterns and even to predict match results to a pretty high accuracy. This suggests that the chat context is an important aspect in Dota 2 games that sometimes can affect and determine the result of the games.

Future Work:

In the future, we need to handle bigger data set and gather more sentimental data. Our suggestion is that people who tend to talk nicely to others get more winning chance and a comfort language environment enable them to communicate better and play better. We still need to put more efforts in cleaning and transforming players' chat content and better performed sentimental test. More analysis will focused on individual player chatting habits. Since single player's chatting may more related with his game performances. While kill/death ratios, gold, experience gain and winning rate can be the criteria for performances assessment and then we can do predictions on single player level. We will investigate more about feature selections, and that can probably be implemented by deleting low information gain features and picking feathers by rank to achieve higher accuracy and processing speed.

Our final goal will be generating a classification system help to match players with similar rank, skills and even chatting habits. For example, rude players can get more chance to play with people like those polite people may want to play more often. I think a will give Dota 2 gaming a better play environment and more game balances. In the end, don't forget! Good Game and Have Fun!

References:

1. Yang, Pu and Roberts, David L.*Knowledge Discovery for Characterizing Team Success or Failure in (A)RTS Games*. http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6633645
2. Conley, Kevin and Perry, Daniel. *How Does He Saw Me? A Recommendation Engine for Picking Heroes in Dota 2*. https://kaggle2.blob.core.windows.net/forum-message-attachments/112112/3944/PerryConley-HowDoesHeSawMeARecommendationEngineForPickingHeroesInDota2.pdf?sv=2012-02-12&se=2016-11-05T02%3A05%3A23Z&sr=b&sp=r&sig=gYZIGKDHEtHo%2F6YhRwMaK9AAplhb6TEoX4hd4DKlnIQ%3D

3. Drachen, Andres. *Skill-Based Differences in Spatio-Temporal Team Behaviour in Defence of The Ancients 2 (DotA 2).* http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7048109
4. Pobiedina, Nataliia. *On Successful Team Formation: Statistical Analysis of a Multiplayer Online Game*. http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6642858
5. Eggert, Christoph. *Classification of Player Roles in the Team-Based Multi-player Game Dota 2*. http://link.springer.com/chapter/10.1007/978-3-319-24589-8_9
6. Dataset: Devin. *Dota 2 Matches*. https://www.kaggle.com/devinanzelmo/dota-2-matches
7. *The International 2016*. Liquipedia Dota 2. http://wiki.teamliquid.net/dota2/The_International/2016
8. A mini map in DOTA 2. https://en.wikipedia.org/wiki/Dota_2
9. Valve analyze about DOTA 2. http://www.gosugamers.net/dota2/news/31911-valve-decides-to-release-the-international-5-statistics-guide
10. Valve own research about DOTA 2 players communication. http://www.eurogamer.net/articles/2013-05-29-valve-explains-how-dota-2s-communication-ban-system-is-training-its-player-base-to-be-more-friendly
11. SnowballStemmer. https://pypi.python.org/pypi/snowballstemmer
12. NLTK http://www.nltk.org/
13. nltk.corpus http://www.nltk.org/book/ch02.html
14. WordCloud https://pypi.python.org/pypi/wordcloud
15. CountVectorizer http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html
16. TfidfTransformer http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html
17. Sentiment Intensity Analyzer http://www.nltk.org/api/nltk.sentiment.html
18. svm http://scikit-learn.org/stable/modules/svm.html
19. GaussianNB http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html
20. RandomForestClassifier http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html