CIS 602 – 02 Special Topics

Project Phase – 3

Final Report (Dota 2 – The Beginner's Guide)

Group Members:

1. Sujoy Kar

2. Hardik Sankhla

3. Hussain Ul Abideen

4. Anurag Singh

Instructed by:

Prof. Maoyuan Sun

Munzner's Table:

System	Dota 2 – the Beginner's Guide
Data Types	Json(database) + API returning Json files (database)
Derived data	Array of match Summary (an object of json files)
Domain	Dota/Steam database
Domain Tasks	API calls
Abstract task	1. Get match history 2. Get match id 3. Get match summary 4. Get hero id 5. Filtering 6. Matching 7. Grouping
View comp.	1. Spacious 2. Area marked in tabular layouts (1 key list) 3. Stacked bar chart, 2 color hues to differentiate the length of the attributes (magnitude of win and loss ratio) 4. Tooltip 5. Highlight
Reduction	Filtering and matching (heavily)
View coord.	Small multiples (1 layout)
Scalability	1. If number of heroes increases, then accordingly visualization (svg) will be scaled.

Introduction:

Dota 2 has become one of the biggest online multiplayer game in the world and thus there has been an increase in the number of new upcoming players. So, for that reason we decided to create this guide for new players. So, to make it our main aim was to create a proper algorithm that would help new players in deciding the top 6 best item choices which will eventually help these players win the game or progress the gameplay in a much simpler manner. We could achieve the basic idea of how to implement our algorithm and present our visualization but while dealing with live data there can be many problems that can happen which we shall discuss in the next section.

Challenges faced and how implementation changes took place:

While trying to handle, the data sets we faced a lot of problems such as:

1. The Defer method that we have implemented in our algorithm is asynchronous and while trying to access such enormous chunk of data results in getting incomplete sets of output. So, that's why we must include a delay to achieve our goal of accessing all data in one try itself.

2. The item ID and index of item array are not equal which created more computations, same was the case for Hero ID which resulted in improper computations for each hero with respect to their respective items being picked.
3. Recently while working on this project, the company, and developers of Dota 2 (valve/steam) decided to add a new patch update to the game which resulted in several changes in the items, hero mechanism, and previously saved data. The changelog of the game can be referred from www.dota2.com/700.
4. While working with the code, there were several instances where we got the same items being used with the exact same number of wins and losses for each hero as the items became global and so were being accessed as same for every hero.
5. When working with around 112 heroes and 281 items together the computation for getting the best 6 item combination becomes too heavy and so ${}^{200}c_6$ combinations were getting tough to be executed without the help of a supercomputer. So eventually we had to come up with other ideas behind the algorithm.
6. Computing these combinations was becoming so difficult that we had to resort to using the individual highest number of bought items with respect to their win/loss ratio and then taking the top 20 items and combining the best out of it to get the proper result.
7. Since valve decided to reset the previously saved data, we were left with little number of items being used in 2-3 days of notice and resulting in so many variations in item win/loss ratio.
8. We had decided to add the tooltip at the end of the data retrieval process as once we would have done with the proper algorithm being executed but with the new update the dota 2 API servers were down to create new sets of datasets for developers to work on so that there would be less load on their servers.

Implementation techniques:

Frontend:

1. **Web Technologies**:

Task Implementation:

We used D3 for our project. To achieve our visualization, we needed to create an svg in which we used a function to call the Hero's and item's images to correspond the connection between each hero and its respective items.

Backend:

1. **Dota 2 Steam API:**

Data Source:

We used Steam API to call all the Data that we need.

JavaScript:

Data cleaning:

1. Firstly, we get a set of match history by this API: http://api.steampowered.com/IDOTA2Match_570/GetMatchHistoryBySequenceNum/v1/?key=332C490A9CE67E401FED8FC2DBEBDCFA, it contains a list of match ids.
2. Then we put a loop and used the match ids to use this API: https://api.steampowered.com/IDOTA2Match_570/GetMatchDetails/v001/?key=332C490A9CE67E401FED8FC2DBEBDCFA&match_id= to get the match summery of all those match id in match history API.
3. We used these match summaries in an Object variable.
4. We used this API: https://api.steampowered.com/IEconDOTA2_570/GetGameItems/V001/?key=332C490A9CE67E401FED8FC2DBEBDCFA to get the list of all the Item id and details.
5. With this API: https://api.steampowered.com/IEconDOTA2_570/GetHeroes/v1/?key=332C490A9CE67E401FED8FC2DBEBDCFA we got a list of all the Heroes id and details.
6. We then made an object such that Item id list has another object ({win, loss}) as the element of every individual Item id object in that list.
7. We then made an object such that Hero id list has another object (New item id list) as the element of every individual hero id object in that list.

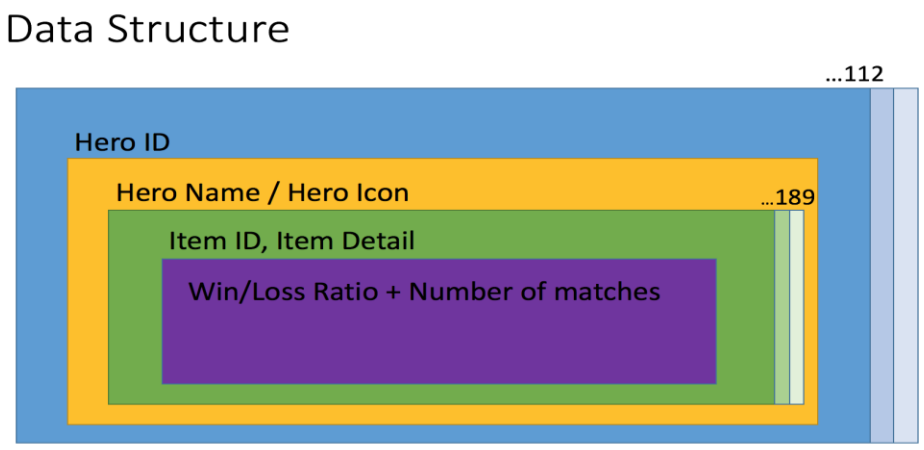
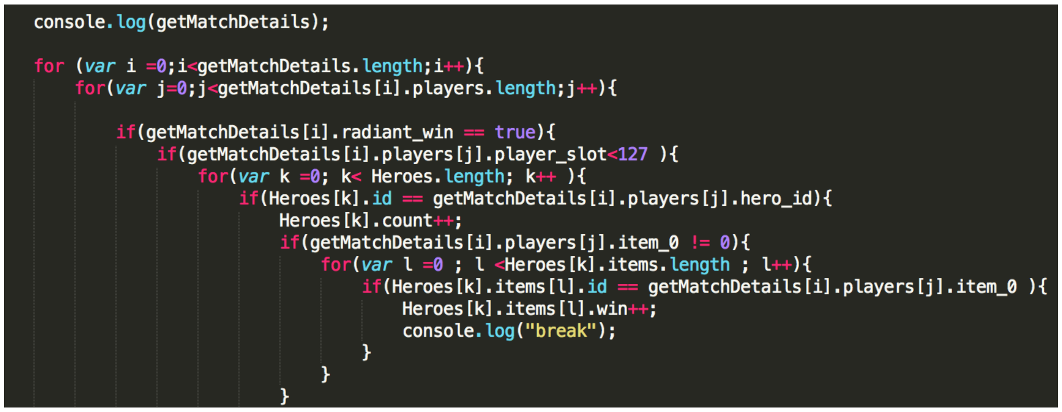


Figure 1: Data Structure

8. Now we have all the necessary required data that we need.

Processing:

1. First, we loop through all the matches, then we loop through all the players, then we collect with which hero did he played, what items did he picked and did he win or lose, now we loop through all the Hero id and the we loop though all the items id and the on the base of data we collected we increment win or loss accordingly.

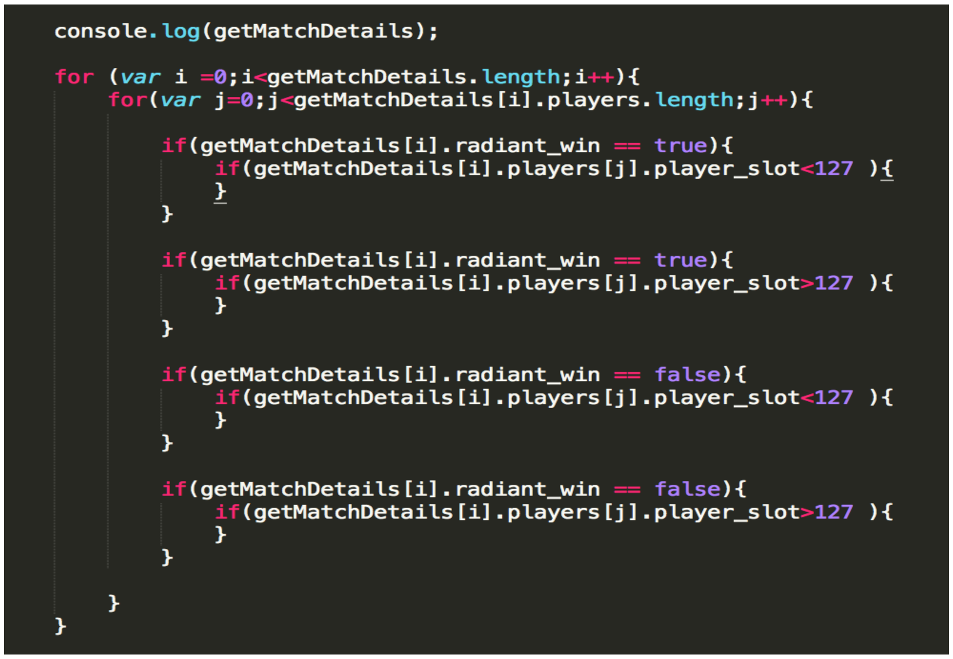


```
console.log(getMatchDetails);

for (var i =0;i<getMatchDetails.length;i++){
    for(var j=0;j<getMatchDetails[i].players.length;j++){

        if(getMatchDetails[i].radiant_win == true){
            if(getMatchDetails[i].players[j].player_slot<127 ){
                for(var k =0; k< Heroes.length; k++ ){
                    if(Heroes[k].id == getMatchDetails[i].players[j].hero_id){
                        Heroes[k].count++;
                        if(getMatchDetails[i].players[j].item_0 != 0){
                            for(var l =0 ; l <Heroes[k].items.length ; l++){
                                if(Heroes[k].items[l].id == getMatchDetails[i].players[j].item_0 ){
                                    Heroes[k].items[l].win++;
                                    console.log("break");
                                }
                            }
                        }
```

Figure 2: Win / Loss Incrementation



```
console.log(getMatchDetails);

for (var i =0;i<getMatchDetails.length;i++){
    for(var j=0;j<getMatchDetails[i].players.length;j++){

        if(getMatchDetails[i].radiant_win == true){
            if(getMatchDetails[i].players[j].player_slot<127 ){
            }
        }

        if(getMatchDetails[i].radiant_win == true){
            if(getMatchDetails[i].players[j].player_slot>127 ){
            }
        }

        if(getMatchDetails[i].radiant_win == false){
            if(getMatchDetails[i].players[j].player_slot<127 ){
            }
        }

        if(getMatchDetails[i].radiant_win == false){
            if(getMatchDetails[i].players[j].player_slot>127 ){
            }
        }

    }
}
```

Figure 3: Win / Loss Condition

2. Now we have win and loss for each item for each hero.
3. Now we sort all the items for every hero on the following basis:

   Value = (Win / (win + loss))* (Win / (win + loss))*Win, In Descending order.
4. Now we take the top 20 most items for each hero and make all the possible 6 item combination and then we chose top 3 combination that had max value.

Screenshot of the visualization:

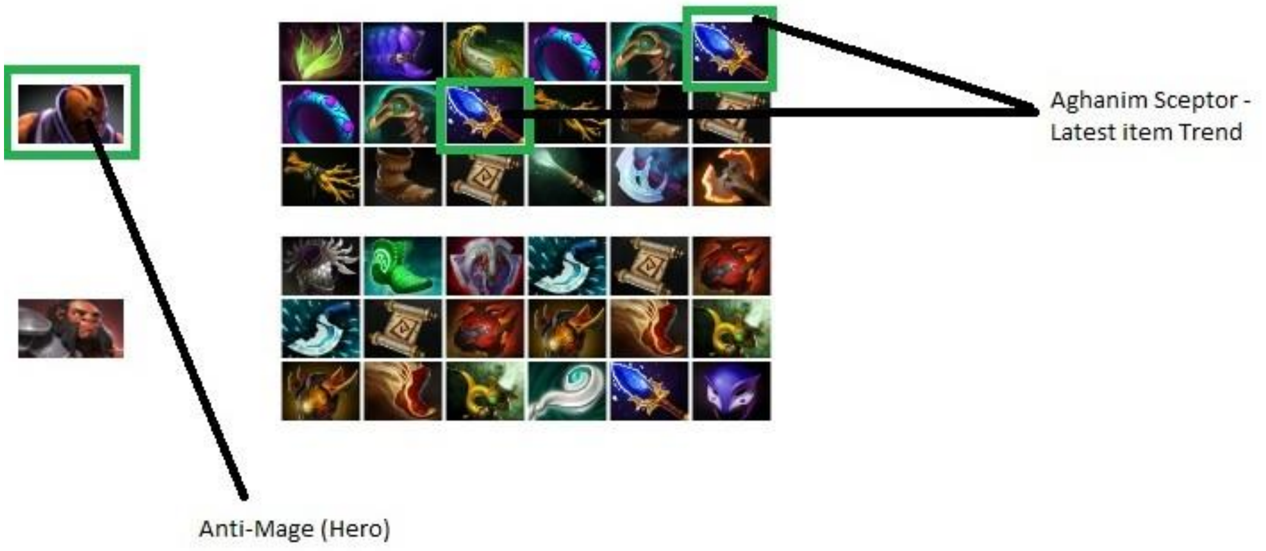


Figure 4: Hero vs Item Build

PROS

1. Our aim was to make it easier for the new players to understand the game mechanics and how each item influences the way a particular hero gets strong in different sets of game.
2. All the set of 6 item combinations shall be preloaded in the background leading to no wait time once the data has been presented online.
3. Visualization should help understand the basic understanding of how these items work hand in hand when used together.

CONS

1. Some items have been included just because there weren't enough items used on a hero and hence items with 0 wins and 0 loss have also been seen in the visualization after the major game update.
2. Due to defer function in D3 being asynchronous it requires a set of delay to execute a big live dataset and so the delay for loading these data could be irritating.
3. Since we were unable to add the tooltip on time it resulted in a stagnant visualization.

Peer Evaluation:

Name	Sujoy Kar	Hardik Sankhla	Hussain Ul Abideen	Anurag Singh
Sujoy Kar	25%	25%	25%	25%
Hardik Sankhla	25%	25%	25%	25%
Hussain Ul Abideen	25%	25%	25%	25%
Anurag Singh	25%	25%	25%	25%