Zewail City of Science and Technology

CIE 427

Big Data Analytics

Mini Project 2 - Analyzing League of Legends

Fatma Moanes NourEl-Din 201700346
Mohamed Abdelrahman 201700030

Special Thanks to Eng. Ahmed Sameh/Elsayed and Eng. Muhammad AlAref

Contents

1 Data

1.1 Data Collection Process

The process of data collection involved collecting around 100k solo/duo ranked matches. The collected matches need to properly match the actual distribution of different ranks. The main logic we used involves collecting 100k different summoners whose ranks are distributed according to the true rank distribution of the game. Afterwards, we collect one match for each summoner, and now we have 100k different matches.
Firstly, we begin by querying the Riot Games API to get different summoner names from all different ranks. In total, we obtain around 1k different summoners and from each rank we obtain a certain number of summoners that would in the end yield an overall rank distribution similar to the real distribution.
The next task is to use this initial list of summoners to get more summoners while maintaining the same ranks distribution. This task is achieved by using a breadth first search approach. Using the BFS terminology, we assume that each summoner in the obtained list forms a node in one level of a search tree. The children of a node are explored/expanded by getting one match for the summoner of the current node, and store 5 different participants among the 10 participants that featured in this match. This tree is explored up to a depth of 3, assuming that depth 0 is the level of the nodes in the initial list of summoners. Finally, we will have more than 100k summoners in the leaves of the tree, and their ranks distribution should be identical to the initial list of summoners.
Now, that we have the list of summoners we need the final task is to store one match for each summoner.

1.2 Data Collection Challenges

The main challenge is that the Riot Games API imposes data rate limits that limits the data transfer to 20 requests every 1 seconds and 100 requests every 2 minutes for a certain API key. To collect 100k matches with such rate limits will take ages, so obviously we had to tune our data collection procedure. Our approach to workaround this issue is to create 30 different Riot Games accounts to get 30 different API keys. We then parallelize the data collection procedure described before by using 30 different threads where each thread uses a different API key. Each thread collects around 4k matches, and it will automatically output the collected data to Google Drive, so we end up with 30 different files containing all the matches we need for the data analysis phase. The data files are combined into one RDD using pyspark union() function.

1.3 Data Problems

The most noticeable problem we faced, was that the API sometimes fails to retrieve the data associated with a certain summoner name. This may be due to the summoner deleting his account or switching between different servers. We dealt with this problem by simply ignoring those erroneous summoners, and collecting others to replace them.