League Analyst

Réalisé par Guillaume Tézenas du Montcel et Romain Blazevic

Scraping

Le Scrapper a pour but de récupérer les données des meilleurs joueurs du jeu « League of Legend » sur le site « League of Graph ». Les données récupérées sont :

Pseudo

Rank

Serveur

Elo (niveau de grade)

LP (score dans ce grade)

Victoire (le nombre de victoire)

%Victoire (le pourcentage de victoire)

Toute ces données seront envoyées dans une base de données MongoDB.

Le Scrapper est codé en Python. Il utilise comme librairie :

Regex

Csv

Requests

Pandas

Html

BeautifulSoup

Le Scrapper contient une Classe « Scrapper » qui possède des méthodes pour récupérer les données.

La classe est divisé en 3 groupes de méthodes distinctes :

Connexion :

Les requêtes à répétition peuvent être détecter par le site est bloqué notre connexion. Des méthodes ont été créé pour palier ceci :

Get_response, init_user_agents

Nettoyage :

Les données récupérées doivent être nettoyer :

remove_white_spaces, remove_white_spaces_split

Récupération de la donnée :

Les données sont récupérées grâce a toutes les méthodes get. Les « extracts_pages » permettent de gérer le défilement des pages.

Classement :

Toutes les données sont ensuite stockées dans un tableau contenant un dictionnaire de joueur.

Serveur Flask

La donnée récupérée est visualisée sur un serveur python Flask. Ce serveur incorpore les éléments NoSQL, comme la base MongoDB, l'indexation Elasticsearch, et les graphiques Matplotlib.

Le dashboard produit permet à l'utilisateur d'avoir la main sur la donnée et de visualiser des informations, notamment la recherche de mots clé ou bien les meilleurs joueurs par serveur.

Graphiques

Les graphiques sur le site ont été réalisé par un programme python.

Les librairies utilisées sont Numpy et Matotlib.

Le programme possède 4 fonctions pour 4 graphes différents :

Chall_by_server_world_pie :

Regroupe le nombre de Challenger par serveur et établi un graphique dit en « fromage » afin de voir facilement quel serveur possède les meilleurs joueurs.

Number_server_pie :

Détermine combien de pourcent le serveur regroupe des joueurs de chaque grade (Challenger / GrandMaster…) sous forme de graphe en fromage.

Number_server_hist :

Détermine combien le serveur regroupe des joueurs de chaque grade (Challenger / GrandMaster…) sous forme de graphe en histogramme.

Chall_by_server_world_hist :

Regroupe le nombre de Challenger par serveur et établi un graphique en histogramme afin de voir facilement quel serveur possède les meilleurs joueurs.

Mongo

Les données sont stockées sous forme de clé-valeur dans une base de données MongoDB.

L'objet "Query_Agent" contient en attribut l'agent de connexion Mongo ainsi que la collection, ce qui permet l'accès facile via les mots clés "self.collection".

Cet objet contient les requêtes nécessaires à l'import de données via CSV, la suppression de données, ainsi que des requêtes.

Les requêtes se branchent directement dans la base et permettent d'alimenter le dashboard.

La requête "top_X_Criteria" permet de sélectionner un nombre N des meilleurs joueurs pour une catégorie choisie. Cette catégorie peut être aussi bien l'Elo, le Serveur, ou autre valeur arbitraire. Le résultat de cette requête est trié sur le Rank, si on ne précise pas d'autre valeur permettant de faire le tri.

Les fonctions "challengers_per_server" et "elo_per_server" sont des requêtes qui alimentent les graphiques Matplotlib.

"challengers_per_server" ne prend pas d'argument et retourne le compte de Challengers par serveur sous forme de liste.

"elo_per_server" prend en entrée le serveur désiré et retourne le nombre de joueurs pour chaque tranche d'Elo, allant de Challenger à Platinum.

Elasticsearch

Le moteur Elasticsearch implémenté est intégré dans l'agent de connexion "Querying_Agent". De la même manière que pour la connexion MongoDB, on peut y accéder via attribut en tapant "self.elastic".

Les fonctions implémentées permettent de créer un index pour un jeu de données, supprimer un index, lister les index présents, et effectuer des recherches par mot clé.

Les recherches s’effectuent sur le nom de l’utilisateur. On peut également spécifier le montant de correspondances que l’on souhaite afficher.