Université Paris Nanterre

Licence MIAGE

Rapport de projet Graphes et OpenData

Détection de communautés sur Twitch

Projet réalisé du 4 janvier 2023 au 8 avril 2023

Membres du groupe

BALBERAN Jeff 40013016
MEHARI Fanuel 40004506

Remerciements

Merci, merci à tous.

Table des matières

1 Introduction

Pour ce projet de graphes et *open data*, nous avons travaillé sur Twitch (Figure 1), le service de *streaming* vidéo en direct. Cette plateforme, qui regroupe, en janvier 2023, 7,5 millions de *streamers* actifs dans le monde ([TWITCHTRACKER]), propose des vidéos en direct qui vont de la diffusion d'une partie de jeu vidéo à des évènements de plus grande ampleur.

Nous avons donc essayé d'y déterminer des communautés parmi les 1000 *streamers* (personnes diffusant des vidéos en direct) avec le plus de *followers* (personnes s'étant abonnées gratuitement à un *streamer*).

FIGURE 1 – Logo de Twitch

2 Environnement de travail

Dans ce tableau (Table 2), nous avons détaillé nos environnements de travail.

	Fanuel MEHARI	**Jeff BALBERAN**
Système d'exploitation	Windows	
Processeur	Intel(R) Pentium(R) CPU N4200	Intel Core i5-6500
RAM	4Go	16Go
Carte graphique	Intel(R) HD Graphics	NVIDIA GeForce GTX 1060 3GB

TABLE 1 – Nos environnements de travail

3 Description du projet et objectifs

3.1 Description du projet

Ce projet nous permet, après avoir récupéré les informations (nom et nombre de *followers*) des 1000 *streamers* avec le plus de *followers* au monde (via le site [SULLYGNOME]), de créer un graphe non orienté où les *streamers* (sommets) sont liés si leur nombre de *followers* en commun dépasse un certain seuil. Ces arêtes ont comme poids le nombre de *followers* communs aux deux extrémités.

3.2 Objectifs

Les objectifs de ce projet sont de faire ressortir des communautés de *streamers* (c'est-à-dire des *streamers* qui ont un nombre significatif de *followers* en commun) et

de pouvoir détecter avec quels autres *streamers*, un *streamer* particulier a le plus de *followers* communs.

4 Bibliothèques, outils et technologies

4.1 Technologies

Pour coder, nous avons choisi le language Python (3.11), qui nous a permis grâce à ses multiples modules de créer notre graphe plus facilement.

4.2 Bibliothèques

Dans ce tableau (Table 4.2), nous avons listé les bibliothèques utilisées dans notre environnement virtuel.

csv
random
networkx
matplotlib

TABLE 2 – Bibliothèques utilisées

4.3 Outils

Nous avons, tous les deux, utilisé Visual Studio Code comme éditeur de texte.

5 Travail réalisé

Tout d'abord, notre fichier **Nettoyage_CSV.py** s'occupe de retirer les informations qui ne sont pas nécessaires pour garder le nom et le nombre de *followers* des 1000 *streamers*.

Ensuite, dans notre fichier **Main.py**, nous avons nos fonctions de création du graphe :

- **charger_graphe()** qui récupère les données du CSV pour construire le graphe.
- **estimer_followers_communs(nb_a, nb_b)** (cf 6.2).
- **detecter_communautes(G)** qui utilise la fonction louvain_communities(G) de la bibliothèque networkx (Table 4.2) pour détecter les communautés puis attribue à chaque sommet une couleur propre à leur communauté.

De plus, nos fonctions **on_node_click(event)** et **afficher_infos_sommet(sommet)** nous permettent d'avoir des graphes interactifs puisqu'elles affichent, lorsque l'on clique sur un sommet, son nom, son nombre de voisins et les noms de ses derniers.

Enfin, les fonctions **afficher_matrice(matrice)** et **initialiser_matrice()** ne sont pas nécessaires au code mais offrent la possibilité d'observer les données dans une structure faite pour les stocker.

Ce tableau (Table 5) récapitule toutes les fonctionnalités.

Nettoyage du CSV
Estimation du nombre de *followers* communs à deux *streamers*
Création du graphe
Détection des communautés
Interactivité avec le graphe
Visualisation des données dans une matrice (pas nécessaire)

Table 3 – Fonctionnalités

Ce tableau (Table 5) récapitule la répartition du travail.

Fanuel MEHARI	**Jeff BALBERAN**
Nettoyage du CSV	
Estimation du nombre de *followers*	
Création du graphe	
Détection des communautés	
	Interactivité avec le graphe

Table 4 – Répartition du travail

6 Difficultés rencontrées

6.1 Récolte des données via l'API

Nous avons, tout d'abord, voulu récolter nos données en utilisant l'API Twitch ([API TWITCH]). Cependant, les méthodes proposées ne nous permettaient pas de récupérer les *streamers* possédant le plus de *followers* ; nous avons donc choisi d'obtenir ces données en téléchargeant un fichier CSV sur le site [SULLYGNOME] qui propose énormément de statistiques sur Twitch.

Par la suite, après avoir utiliser l'API dans notre fichier Nettoyage_CSV.py pour récupérer les ID des *streamers*, nous nous apprêtions à récolter les données sur les *followers* pour effectuer nos comparaisons entre chaque *streamer* lorsque nous nous rendîmes compte que l'API nous proposait seulement de récupérer les données individuelles de chaque *follower*.

Compte tenu du très grand nombre de *followers* (par exemple, le premier *streamer*, Ninja, en a 18 millions), nous n'avions, comme solutions, plus que le stockage sur une base de données ou la possibilité d'essayer d'estimer le nombre de *followers* en commun entre deux *streamers*. La première option nécessitant d'effectuer une quantité excessive d'appels à l'API sur une trop grande période de temps puisque nous devrions récolter des millions de lignes avec une limite de 800 appels par minute, il ne nous restait plus que la possibilité d'obtenir des données qui ne seraient pas exactes mais qui se rapprocheraient de la réalité.

6.2 Estimation du nombre de *followers* communs

Pour estimer le nombre de *followers* communs à deux *streamers*, nous avons donc choisi la méthode d'échantillonage aléatoire simple et la formule de Cochran ([WIKIPEDIA]). Néanmoins, trouver le moyen d'obtenir des valeurs régulières entre les exécutions de notre code Main.py s'est avéré difficile. Nous avons donc dû nous résoudre à utiliser une formule, que nous ne connaissions pas, proposée par [CHAT GPT] (Figure 2) pour obtenir des valeurs plus raisonnables.

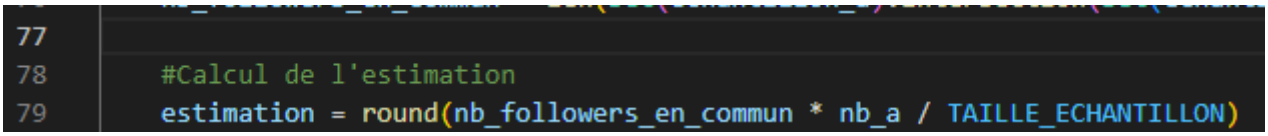

FIGURE 2 – Formule d'estimation

7 Bilan

7.1 Conclusion

Malgré les difficultés de récolte et d'estimation des données, ce projet nous a permis, à partir d'un simple fichier CSV de 1000 lignes, de pouvoir créer, manipuler et interpréter un graphe qui fait ressortir des communautés qui, initialement, n'étaient pas visibles. Le graphe obtenu sera brièvement interprété dans l'exemple d'exécution du projet (A).

7.2 Perspectives

Parmi les pistes d'amélioration, on peut relever la nécessité de trouver une source de données plus fiable. De plus, l'exécution du fichier Main.py prenant un temps considérable pour traiter 1000 streamers, une tentative d'optimiser ce code constituerait aussi une amélioration.

8 Webographie

Références

[PYTHON] https://docs.python.org/fr/3/

[NETWORKX] https://networkx.org/documentation/stable/reference/introduction.html

[API TWITCH] https://dev.twitch.tv/docs/api/

[CHAT GPT] https://openai.com/blog/chatgpt

[WIKIPEDIA] https://fr.wikipedia.org/wiki/%C3%89chantillon_(statistiques)

[SULLYGNOME] https://sullygnome.com/

[TWITCHTRACKER] https://twitchtracker.com/statistics/active-streamers

9 Annexes

Annexe A : Exemple d'exécution du projet

Annexe B : Manuel utilisateur