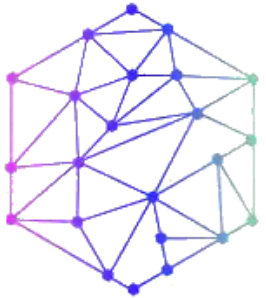
Graph Mining

Carl Rizk, David de la Hera, Noam Benitah



Sommaire


twitch

A) Objectifs

1. Retrouver les communautés linguistiques

2. Identifier les métriques plus remarquables

3. Comparer les résultats

B) Description du graph

Nature du graphe

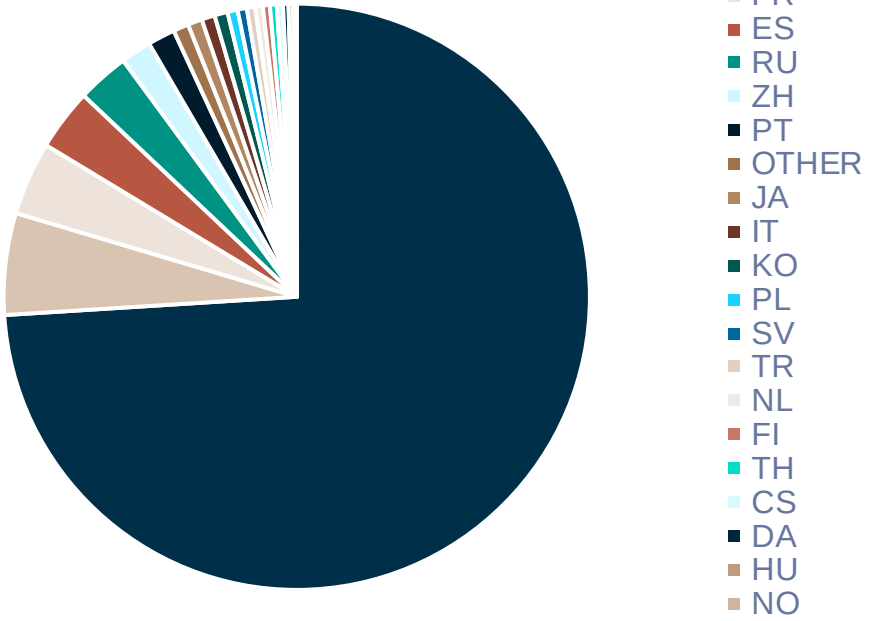
- Type de graphe : non orienté
- Domaine d'application : service de streaming (Twitch)

Structure du graphe

- Nœuds : Utilisateurs de Twitch
- Arêtes : Following mutuelle
- Labels: 21 langues différentes

Taille originale

- 168 114 Nœuds
- 6 797 557 Arêtes

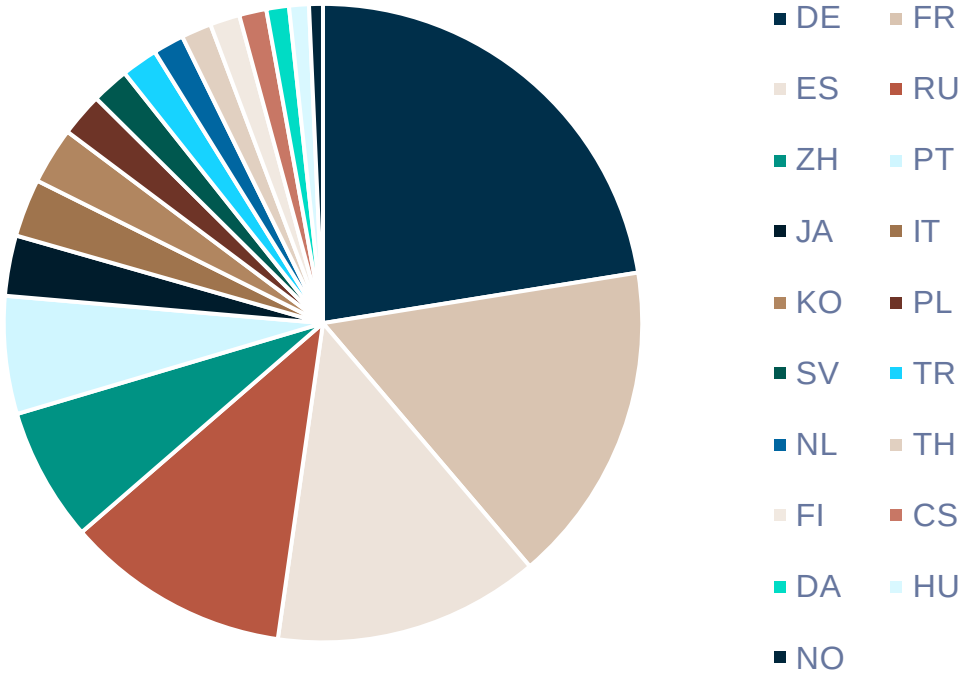


Lien du graph: https://snap.stanford.edu/data/twitch-social-networks.html

B) Description du graph

Graph finale

- 41 265 Nœuds (~24.5%)
- 657 892 Arêtes (~9.7%)
- 19 langues
- Diamètre de 10 nœuds
- Rayon de 5 nœuds



C) Algorithme utilisé

Initialise chaque nœud avec un label unique
Pour n itérations:
    Pour chaque nœud x:
        label(x) = label le plus courant* dans le voisinage de x

*Possibilité d'ajouter des poids aux arêtes

✓ Complexité O(|E|)

✓ Paramétrable

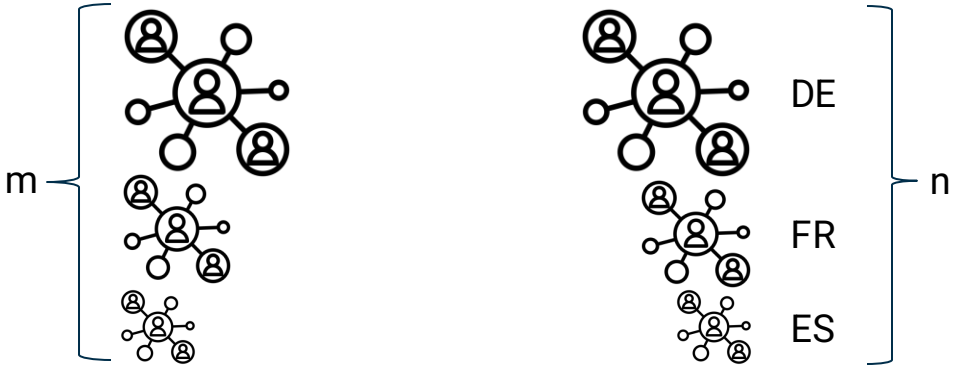
× Algorithme probabiliste

× Problématique si les clases ne sont pas équilibrées

Raghavan, Usha Nandini, Réka Albert, and Soundar Kumara. "Near linear time algorithm to detect community structures in large-scale networks." Physical Review E 76.3 (2007): 036106.

D) Méthode

- Générer les communautés grâce a l'algorithme de Label Propagation
- Associer les communautés calculés aux communautés réelles:



Pour chaque com des min(m, n) plus grandes communautés calculées:
Trouver la communauté réelle ayant la plus grande intersection avec com
(Une communauté réelle ne peut être associé qu'à une seule communauté calculée)

D) Méthode

- Générer les communautés grâce a l'algorithme de Label Propagation
- Associer les communautés calculés aux communautés réelles
- Calculer la précision des associations:

CC 

$$Score_{association} = \frac{|CC \cap CR|}{|CC \cup CR|}$$

 CR

$$Score = \frac{\sum |CC \cap CR|}{nbre\_noeuds}$$

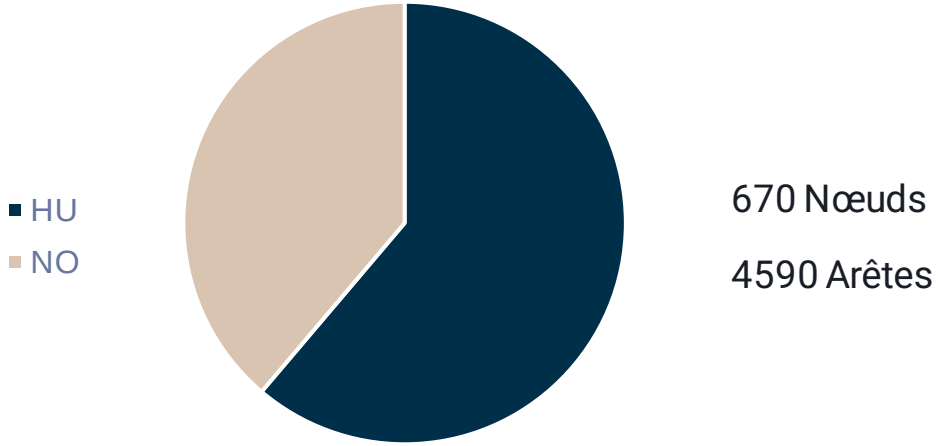
E) Analyse et résultats

Premier Test: Exécuter l’algorithme avec des poids égale a 1

Poids	Score (5 exécutions)			
	Moyenne	STD	Min	Max
1	91.73%	3.2%	87.46%	94.72%

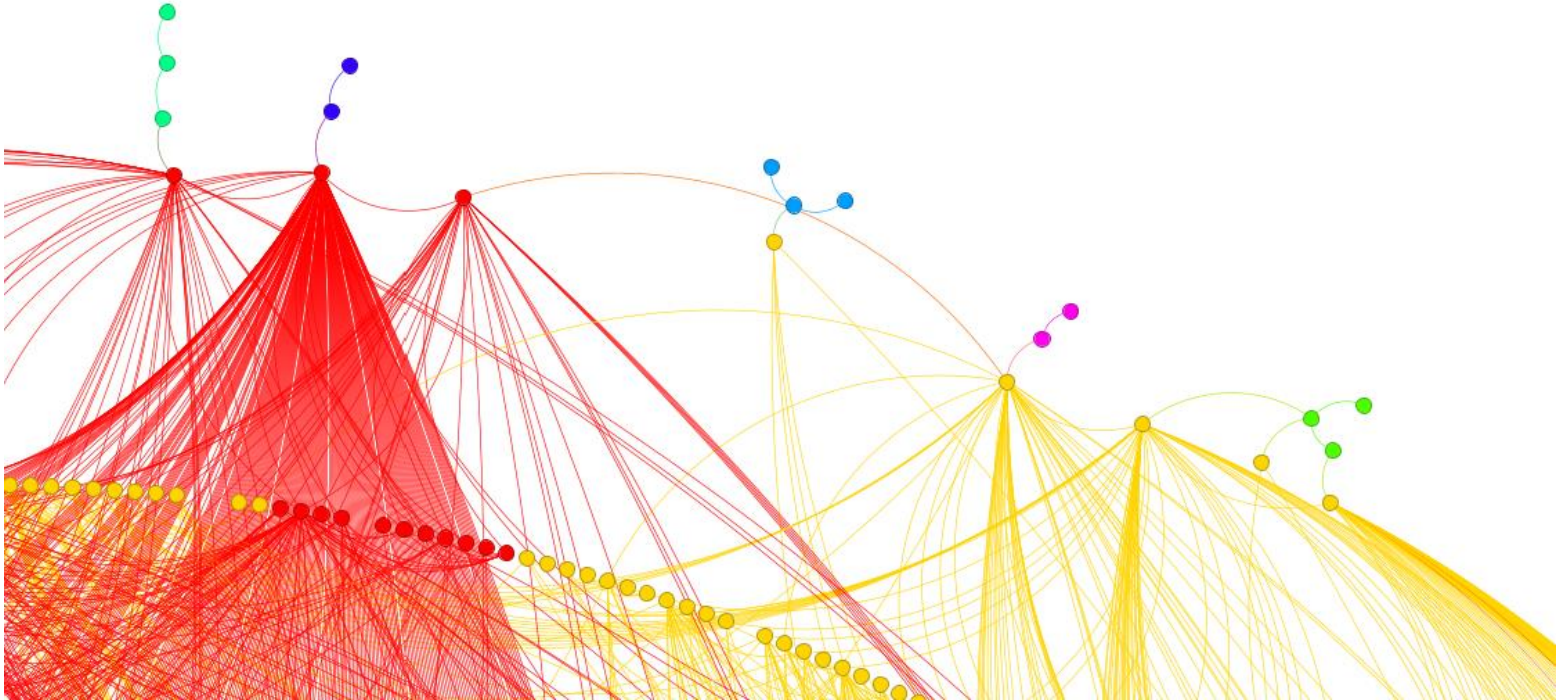
Comment Améliorer

E) Analyse et résultats



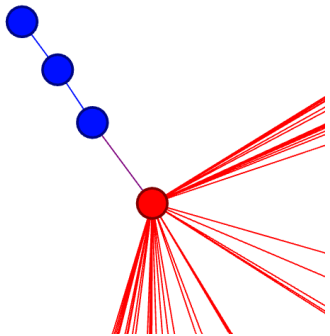
Poids	Score (500 exécutions)			
	Moyenne	STD	Min	Max
1	97.91%	1.65%	80.14%	99.10%

E) Analyse et résultats



E) Analyse et résultats

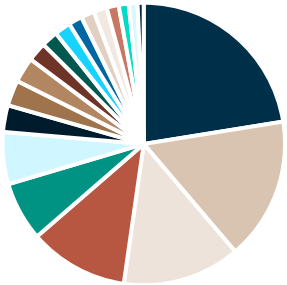
Poids	Score (500 exécutions)			
	Moyenne	STD	Min	Max
1	97.91%	1.65%	80.14%	99.10%
Min degré	99.39%	0.83%	81.04%	99.85%



E) Analyse et résultats

Poids	Score (500 exécutions)			
	Moyenne	STD	Min	Max
1	97.91%	1.65%	80.14%	99.10%
Min degré	99.39%	0.83%	81.04%	99.85%
Min Betweeness Centrality	99.76%	1.7%	61.20%	99.85%

E) Analyse et résultats



DE FR ES
RU ZH PT
JA IT KO
PL SV TR
NL TH FI
CS DA HU
NO

Poids	Score (5 exécutions)			
	Moyenne	STD	Min	Max
1	91.73%	3.2%	87.46%	94.72%
Min degré	90.51%	3.77%	86.17%	96.09%
Min Betweeness Centrality	Trop long a calculé			

E) Analyse et résultats

Poids	Score (5 exécutions)			
	Moyenne	**STD**	**Min**	**Max**
1	91.73%	3.2%	87.46%	94.72%
Min degré	90.51%	3.77%	86.17%	96.09%
Min Betweeness Centrality	Trop long a calculé			
Min Degree Centrality	91.15%	4.17%	86.41%	97.56%

F) Conclusion

Application Réelle

1. Retrouver les communautés linguistiques

2. Identifier les métriques plus remarquables (Problème de complexité)


twitter

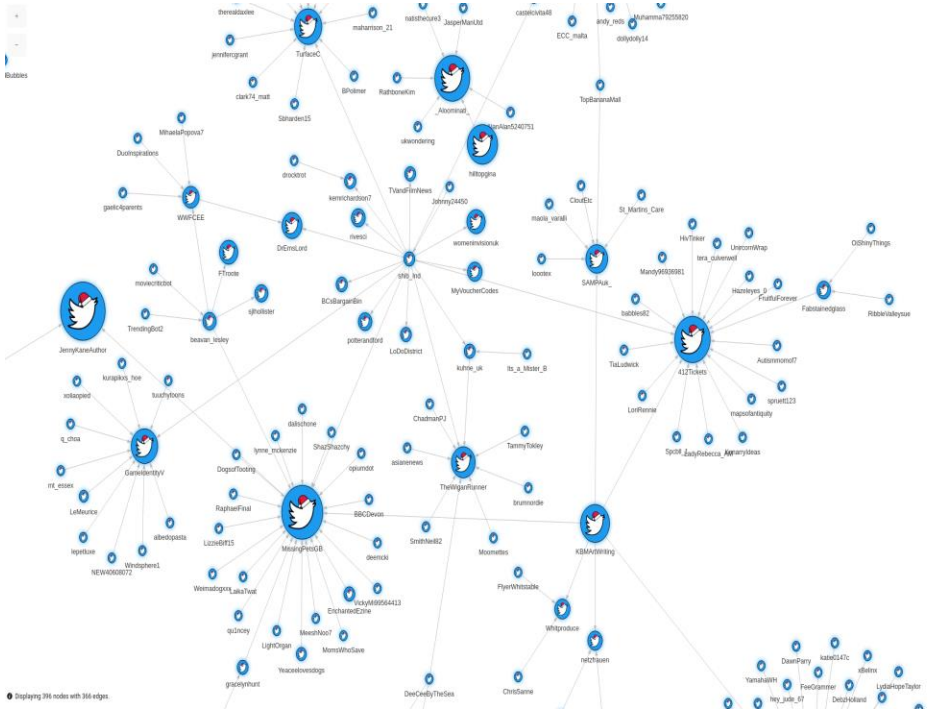
A) Objectifs

1. Identifier les utilisateurs influents

2. Comparer les modèles de diffusion

3. Optimiser les stratégies de diffusion

B) Description du graph de twitter



Lien du graph: https://snap.stanford.edu/data/ego-Twitter.html

Nature du graphe

- Type de graphe : orienté non pondéré
- Domaine d'application : réseaux sociaux (Twitter)

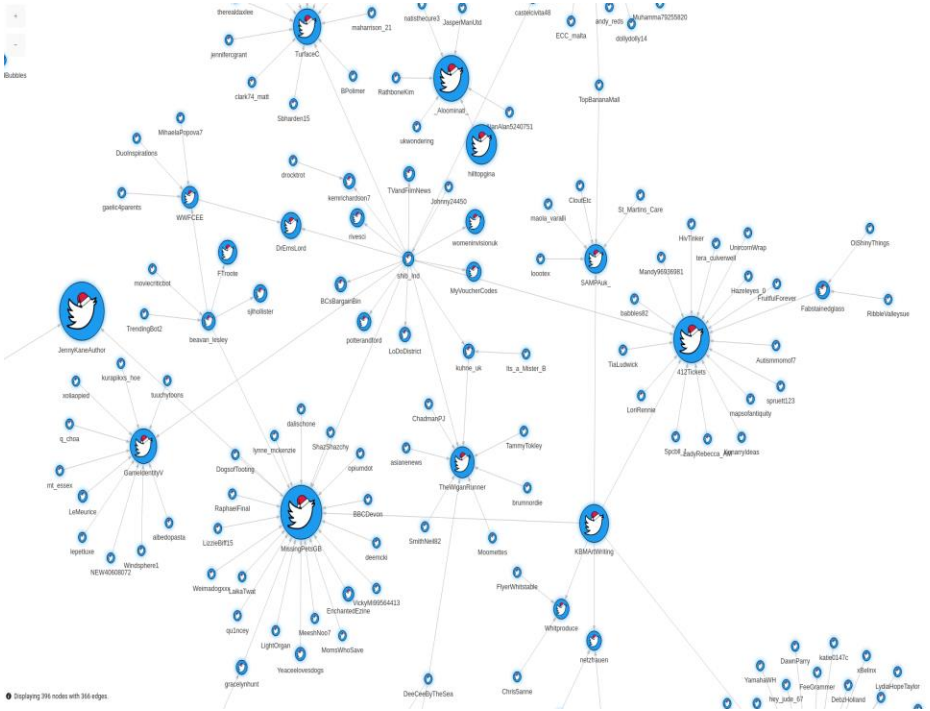
Structure du graphe

- Nœuds : Utilisateurs de Twitter
- Arêtes : Relations entre les utilisateurs (abonnements et abonnés)

Taille originale

- 81 306 Nœuds
- 1 768 149 Arêtes

B) Description du graph de twitter

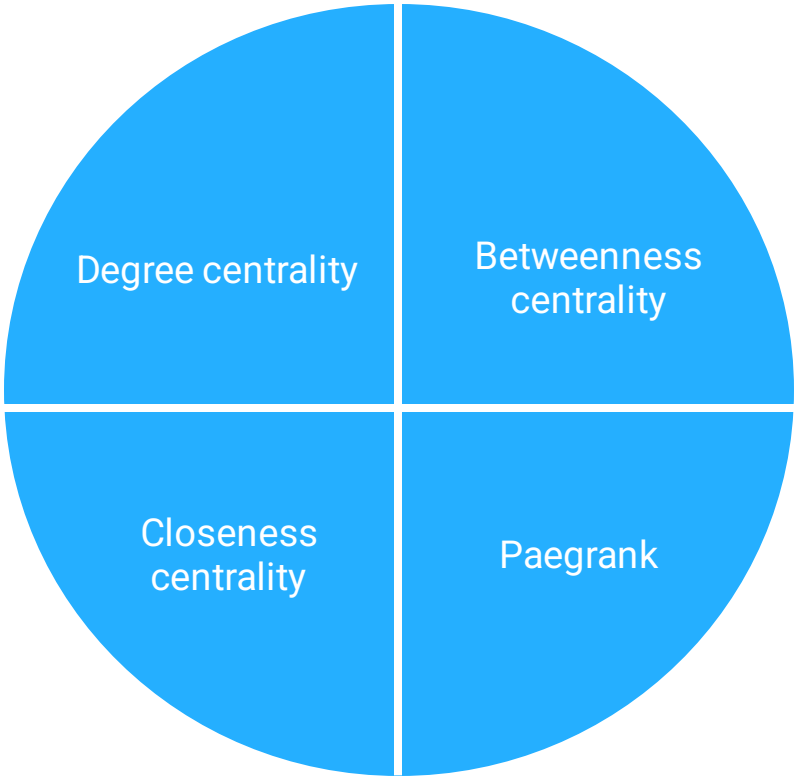


Structure du sous-graph

- Type de graphe : orienté non pondéré
- Reduction du graphe a partir du nœud avec la plus grande centralité (dégrée). BFS jusqu'à 3000.
- 3 000 Nœuds (~3,69%)
- 75 413 Arêtes (~4,27%)

C) Étude centralité

- Étude de plusieurs mesures de centralité
- Trouver les nœuds les plus influents
- Améliorer le modèle



D) Independant cascade - algorithm

→ Modèle de diffusion pour observer quels utilisateurs sont les plus influents

Étapes

- Choisir K utilisateurs (influents) à activer au départ
  - → On utilisera les utilisateurs avec les meilleures mesures de centralités
- Chaque nœud actif u peut activer un de ses voisins v qui n'est pas activé avec une probabilité $1/dv^{-}$

→ **Nous allons comparer les diffusions en fonction de l'ensemble de départ des nœuds activés**

D.2) Independant cascade - Résultats

On étudie le nombre de voisin actif à l'issu de l'algorithme en modifiant certains paramètres :

- La méthode de centralité utilisés pour sélectionner les nœuds de départ activés
- Le nombre de nœuds activés au départ (entre 5 et 40)

→ On effectue 10 simulations à chaque fois en prenant la moyenne de nœuds activés

→ Les tests ont été effectués sur un sous graph de 3000 nœuds pour être réalisable en un temps raisonnable

	degree_centrality	betweenness_centrality	closeness_centrality	pagerank
5	853.1	933.2	704.9	703.7
10	857.0	1062.8	717.5	691.3
20	935.8	1106.9	803.2	809.4
40	1216.9	1212.4	897.0	881.5

E) Linear treshold - algorithm

Différences avec Independant cascade

→ Un nœud est activé si la proportion de ses voisins actifs dépasse son seuil d'activation

→ Le seuil est généré aléatoirement pour chaque nœud et à chaque étape de la simulation.

Chaque nœud a un seuil d'activation aléatoire qui détermine s'il s'active

≠

Chaque nœud a une probabilité de transmission sur chaque arête.

E.2) Linear tresholds- Résultats

Nous avons effectué la même étude que sur Independant Cascade afin de les comparer

	degree_centrality	betweenness_centrality	closeness_centrality	pagerank
5	2118.1	2055.2	2140.8	2165.9
10	2126.5	2052.2	2177.6	2168.9
20	2172.8	2133.5	2193.2	2225.2
40	2195.9	2149.4	2266.0	2259.1

F) Influence maximization problem - Algorithm

→ Améliorer les résultats d'influence en choisissant de meilleurs nœuds

→ Recherche des nœuds de départ à activer les plus performants

Étapes

- Commencer avec un ensemble de nœuds de départ vide
- Ajouter le nœud qui maximisera le nombre de nœuds activés si on le place dans l'ensemble de départ
- Itérer jusqu'à obtenir le nombre de nœuds souhaités dans l'ensemble de départ

F.2) Influence maximization problem – Independant Cascade

Les tests ont été effectués sur un sous graph de 1000 nœuds pour être réalisable en un temps raisonnable

	degree_centrality	betweenness_centrality	closeness_centrality	pagerank	influence_maximization
5	632.5	627.3	630.6	628.0	627.0
10	656.7	637.3	627.2	640.6	None
20	661.6	663.3	669.0	652.0	None
40	680.6	684.7	679.0	680.6	None

F.2) Influence maximization problem – Linear treshold

Les tests ont été effectués sur un sous graph de 1000 nœuds pour être réalisable en un temps raisonnable

	degree_centrality	betweenness_centrality	closeness_centrality	pagerank	influence_maximization
5	410.2	413.7	447.7	444.5	401.5
10	447.8	433.9	456.2	464.6	None
20	465.2	468.5	485.8	484.0	None
40	516.4	507.4	517.8	515.7	None

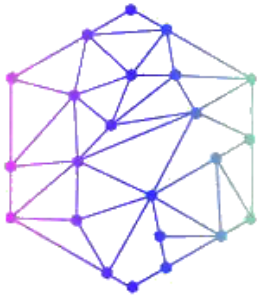
G) Conclusion

Application Réelle

Retrouver les meilleurs « influencer »

Diffuser les campagnes publicitaires à travers des personnes qui ont une plus grand diffusion

MERCI !



Lien du Git : https://github.com/carlrizk/CS-GRAPH