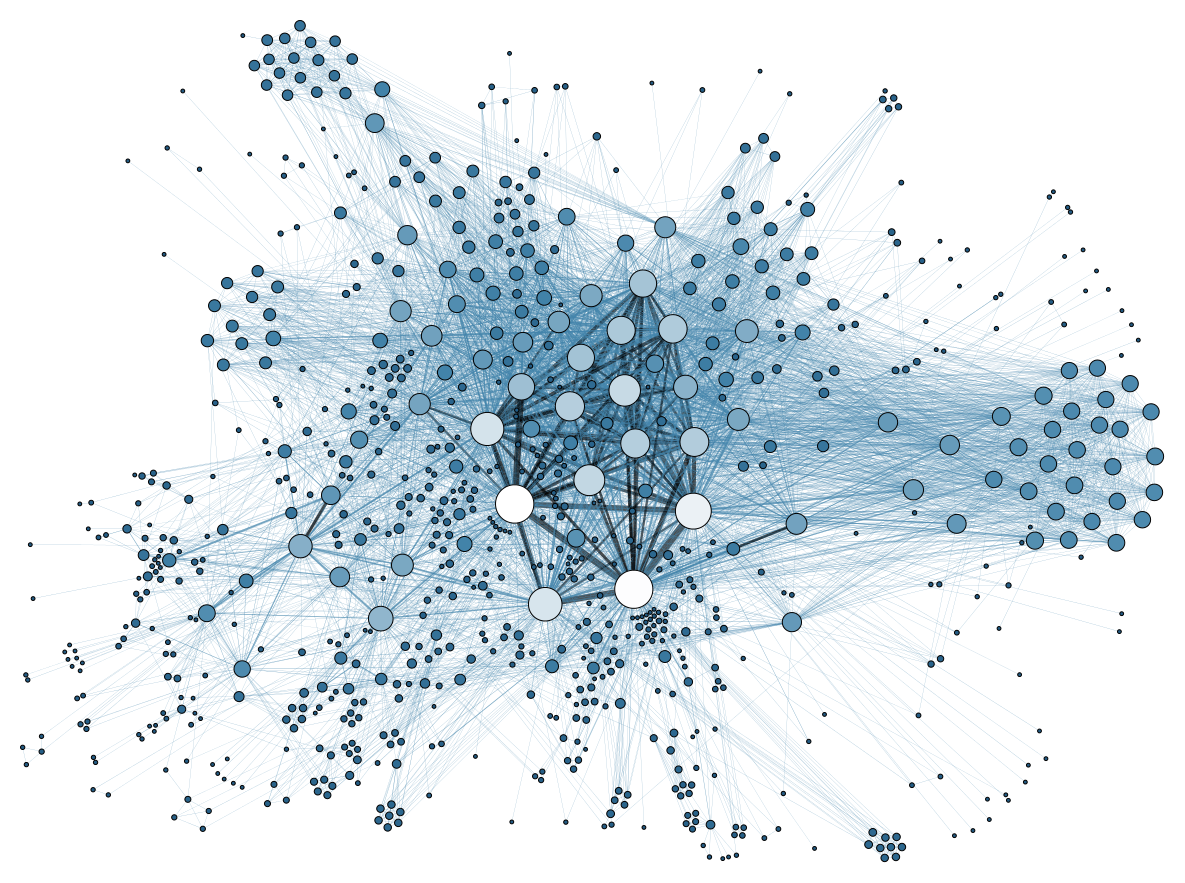
Erwan MARTIN ING3 DS

Joshua BONACORSI

**RAPPORT ANALYSE DE RÉSEAUX SOCIAUX**

**à destination de monsieur Rushed KANAWATI**



**Sommaire**

1. Présentation du sujet
2. Traduction du problème sous forme de graphe
3. Solution proposée

A. Problème des graphes bipartites

B. Projection du graphe biparti vers un graphe unipartite

C. Problème des projections

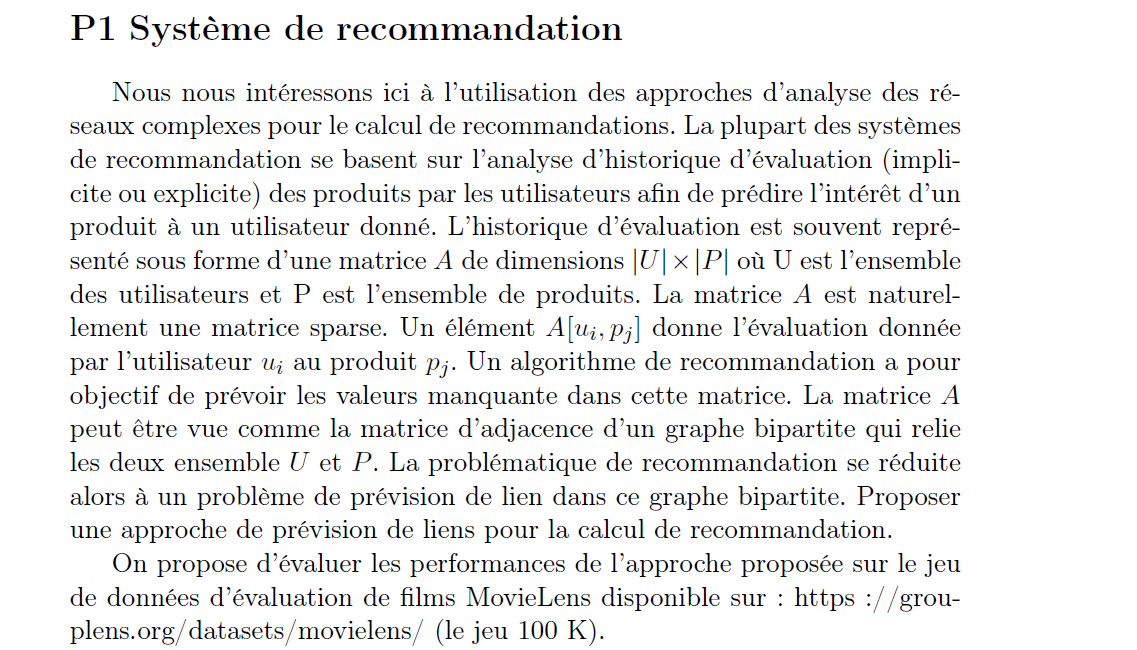
D. Choix de l’algorithme de détection de communauté

E. Résultat de la solution proposée

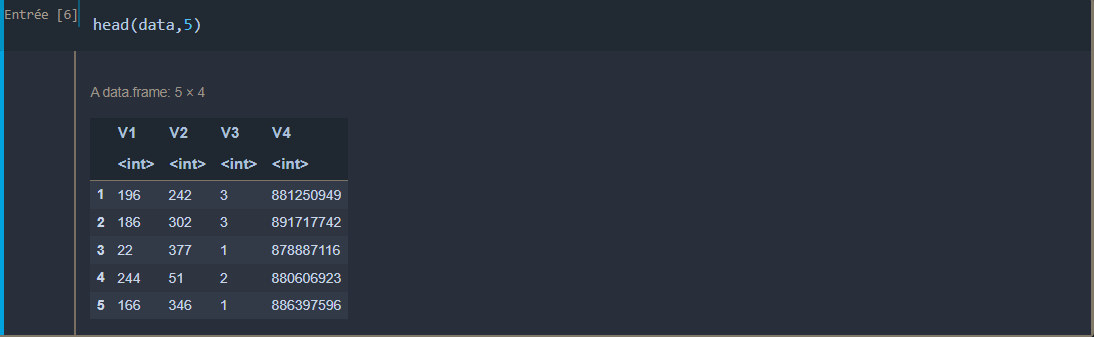
1. Test
2. Conclusion
3. Bibliographie

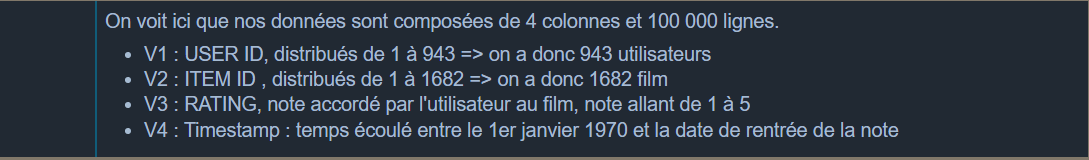
I. Présentation du sujet

Énoncé :

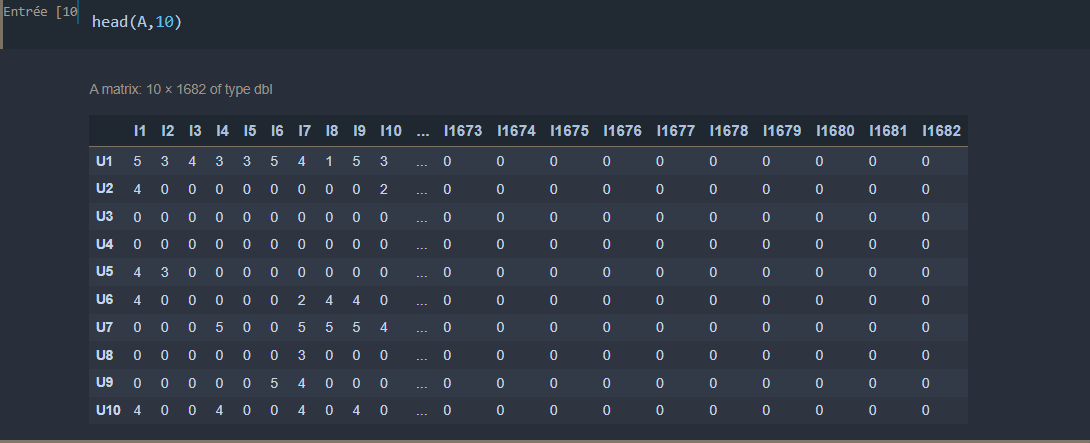


Approche 3 : Application des algorithmes de détection de communautés

Nous possédons un jeu de données de 100 000 lignes et 4 colonnes V1, V2, V3 et V4



On peut à partir de ces données créer la matrice A évoquée dans l’énoncé.



On remarque comme attendu que cette matrice est creuse, on constate 94% de valeurs manquantes (donc de 0).

L’objectif de ce projet est de prédire ces valeurs manquantes grâce à la détection de communautés.

II. Traduction du problème sous forme de graphe

Finalement la matrice A correspond à la matrice d’incidence entre les utilisateurs et les produits (les films ici). On peut alors directement créer un graphe à partir de cette matrice grâce à la fonction *graph\_from\_incidence\_matrix* sur R. On obtient le graphe suivant :

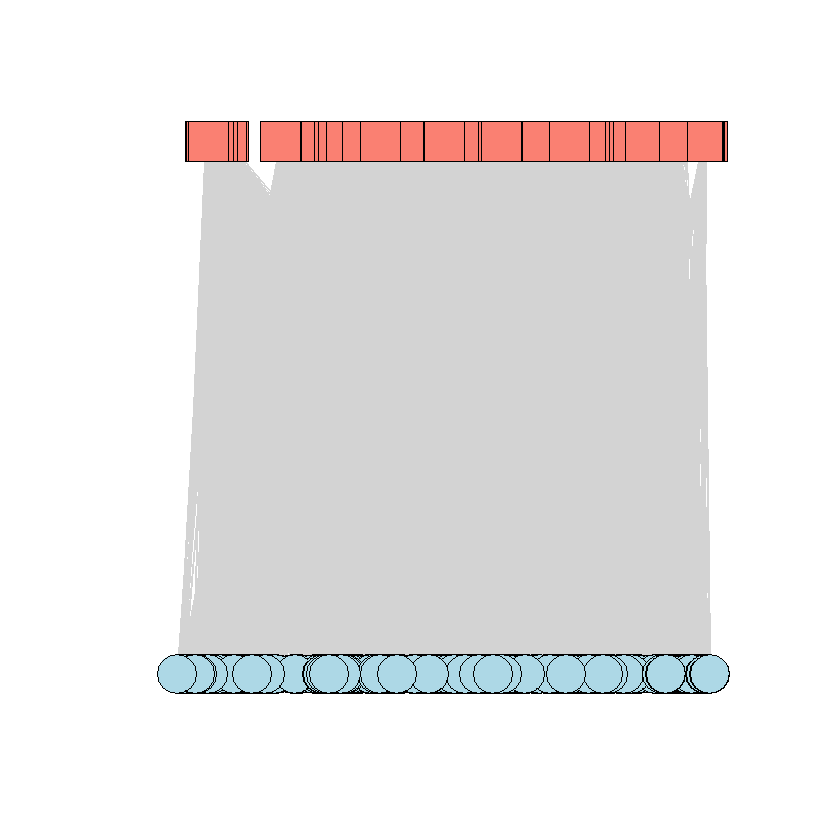


Figure - Graphe représentant les liens entre les utilisateurs et les films

On a bien un graph qui représente les utilisateurs (carré saumon) et les films (cercle bleu) Chaque utilisateur est relié à chaque film qu'il a évalué, l'arête faisant le pont a pour poids la note de l'utilisateur sur le film associé.

Exemple : L’utilisateur U1 note le film I1 3/5, alors il aura un lien U1-I1 de poids 3.

On cherche donc à prédire le poids des liens manquant à l’aide de la détection de communautés.

III. Solution proposée

Pour répondre au problème, nous allons essayer de détecter des communautés et « uniformiser » les valeurs dans une même communauté ce qui veut dire que nous allons utiliser les données disponibles dans cette communauté pour prédire les données manquantes de cette communauté. Cependant un problème se pose.

1. Problème des graphes bipartites

La détection de communauté que nous avons étudier en cours n’est pas adapter au graphe bipartite. En effet, dans toutes les méthodes vues en cours, on travaille sur des graphes « unipartites » c’est-à-dire que tous les nœuds sont de même nature et il n’y a aucune restriction de lien sur le graphe.

Ce n’est pas le cas ici, nous possédons 2 types de nœuds et 2 nœuds du même type ne peuvent pas posséder de lien.

Après une recherche approfondie, plusieurs solutions nous étaient proposées mais la plupart était assez éloigné du cours. On a donc opté pour l’option qui nous permettait d’exploiter le cours au maximum : la projection.

1. Projection du graphe biparti vers un graphe unipartite

L’idée de cette projection est de se ramener au type de graphe que l’on sait exploiter grâce au cours, la projection fonctionne de la manière suivante :

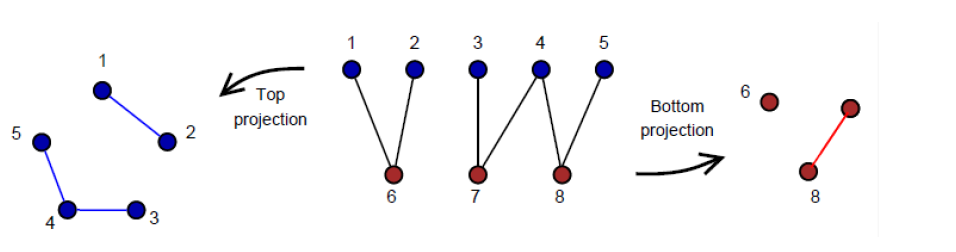


Figure - Schéma du fonctionnement d'une projection d'un graphe bipartite

L’idée est de regarder pour 2 nœuds du même type le nombre de nœud de l’autre type qu’ils ont en commun. On peut choisir de créer une projection pondérée dont chaque lien aura pour poids le nombre de nœud (de l’autre type) en commun.

Ainsi, dans notre cas, on peut créer 2 projections, une en se concentrant sur les utilisateurs et l’autre en se concentrant sur les films.

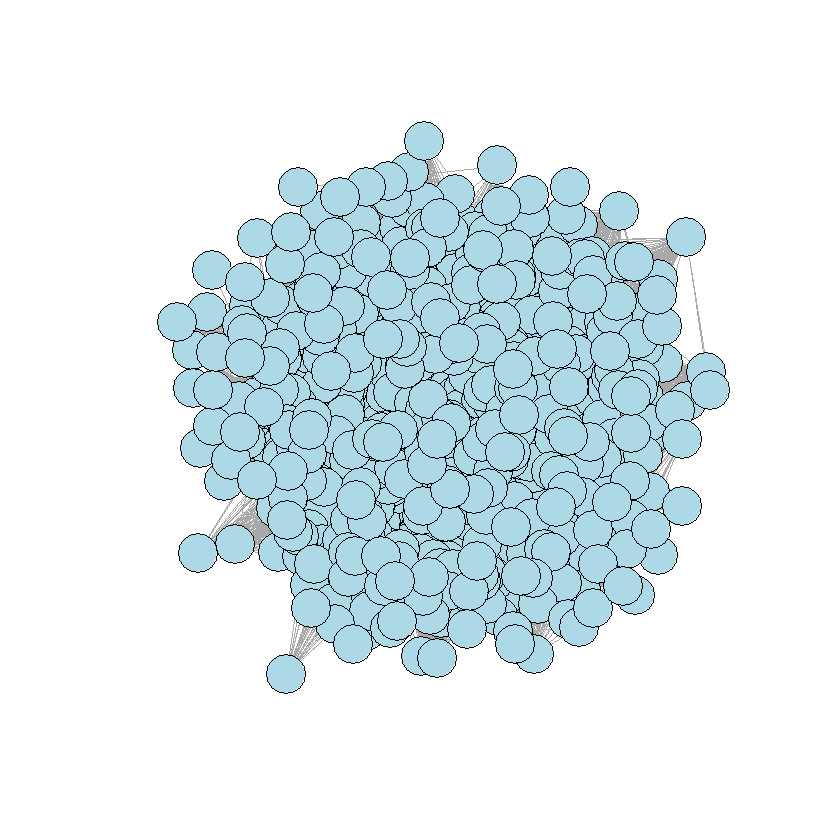


Figure - Projection du graphe bipartite par rapport aux films

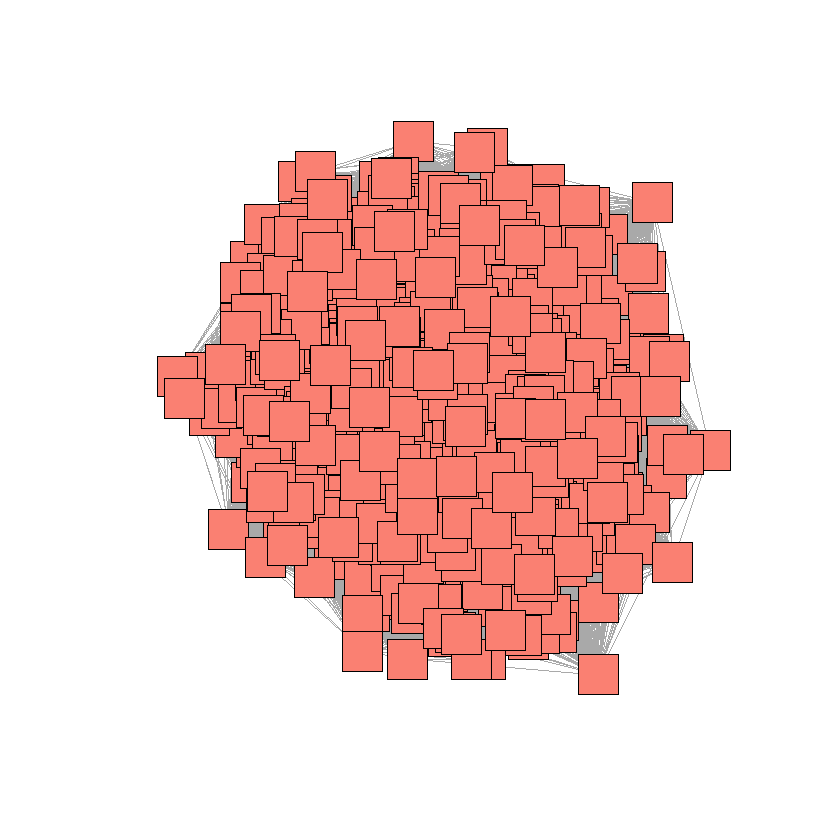


Figure - Projection du graphe bipartite par rapport aux utilisateurs

1. Problème des projections

Le problème des projections, c’est qu’elles ne prennent pas en compte les poids (donc les notes dans notre cas) du graphe initial. Ainsi, on perd l’information qui nous est vitale dans notre problème. Pour palier à cela, nous avons imaginé l’astuce suivante : Traiter chaque « niveau » de poids indépendamment.

Cela veut dire qu’au lieu de travailler sur notre graphe bipartite initial, nous allons créer 5 sous graphe qui représenteront chacun les liens pour une note i.

En effet, dans notre problème il y a 5 notes possibles allant de 1 à 5 on aura donc 5 graphes.

Exemple :

Graphe 1 : Sous graphe du graphe initial représentant uniquement les arêtes de poids égal à 1.

Ainsi, en travaillant sur le graphe 1 par exemple, nous pourrons identifier les utilisateurs ayant les mêmes goûts sur les films qu’ils n’aiment pas.

On aura donc à la fin 5 groupes (un groupe pour chaque graphe) d’ensemble de communautés.

Pour arriver à ce résultat, il faut choisir un algorithme de détection de communauté adapté à notre problème.

1. Choix de l’algorithme de détection de communautés

Pour le choix de l’algorithme, on s’est limité aux algorithmes vus en classe :

* Louvain
* Walktrap
* Edge-betweenness

Après quelques tests sur les différentes projections, il était clair que le meilleur algorithme des trois pour notre problème était l’algorithme Louvain. En effet, ce dernier avait systématiquement la modularité la plus élevée des trois.

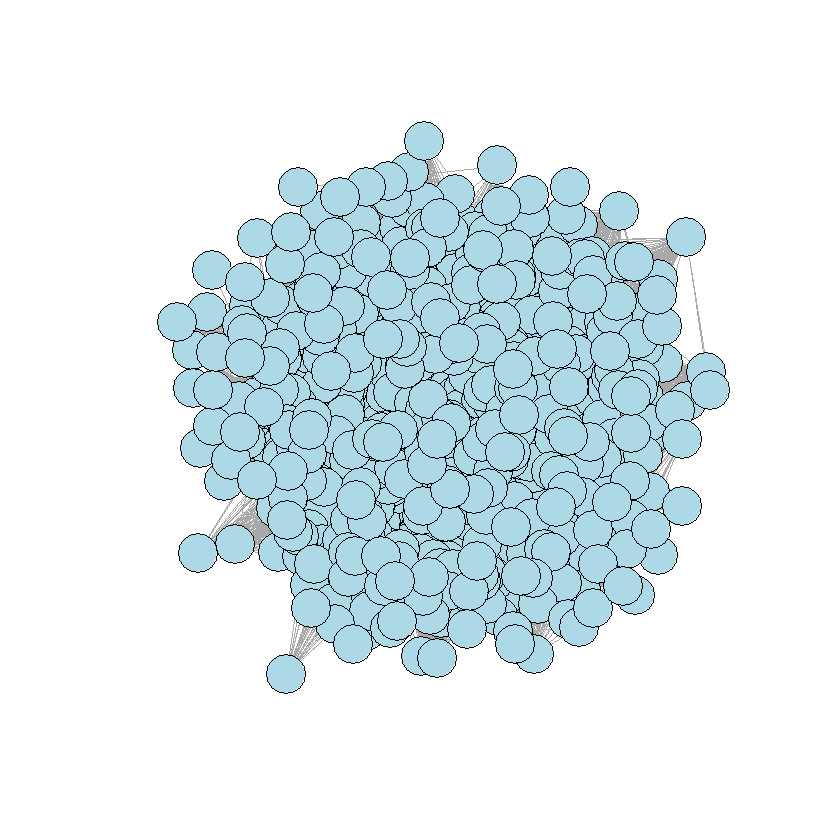
C’est donc cet algorithme que nous utiliserons par la suite.

1. Résultats de la solution proposée
2. Division du graphe bipartite initial en 5 sous graphe.

On commence tout d’abord par diviser le graphe bipartite initial en 5 sous graphe comme décrit dans la partie C.

1. Projection des 5 graphes sur les utilisateurs

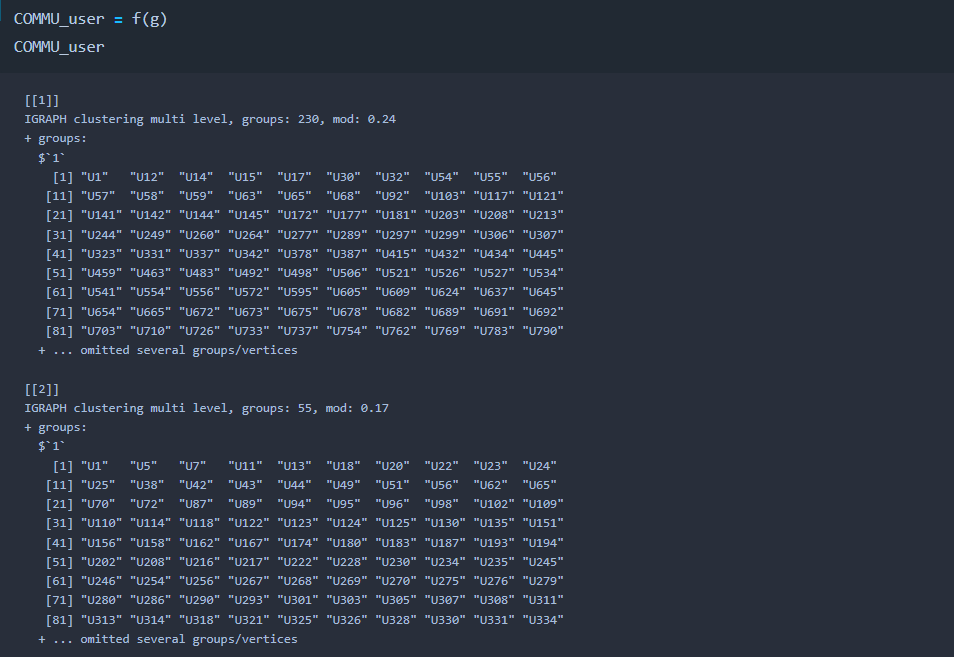
Dans un premier temps, on va projeter chacun de ces graphes sur les utilisateurs.



On possède donc 5 graphes représentant les liens entre les utilisateurs pour une note donnée.

1. Détection de communautés dans chaque projection

On applique l’algorithme de Louvain sur chacune de ces projections. On obtient alors un ensemble de communautés pour chacune des projections :



1. Regroupement des communautés de chaque projection

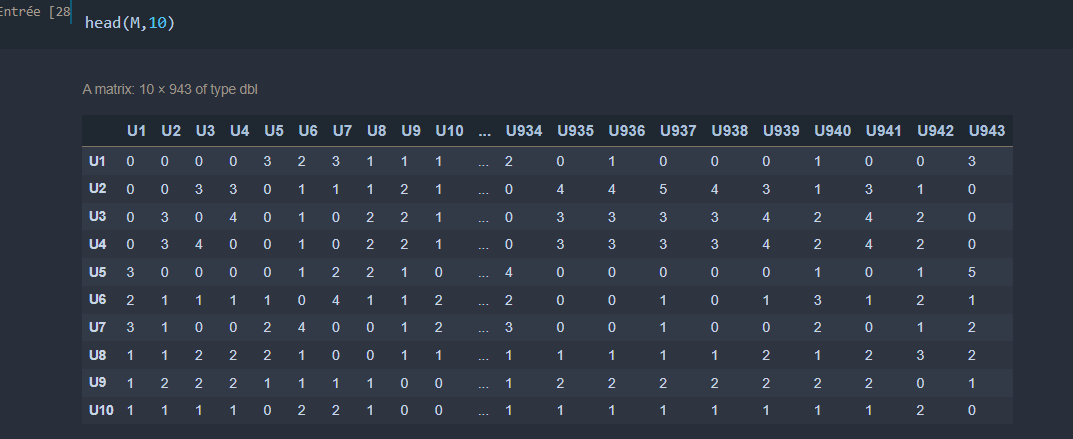
Actuellement, nous avons des communautés propres à chaque sous-graphe, c’est-à-dire des communautés propres à une note donnée. Or ce que l’on cherche à trouver ce sont des communautés « globales » c’est-à-dire des communautés qui regroupent les utilisateurs ayant les mêmes goûts en termes de films que ce soit des films qu’ils aiment ou qu’ils n’aiment pas.

Il faut donc rassembler les résultats obtenus précédemment.

On va donc créer une matrice M qui va parcourir ces groupes de communautés et ajouter +1 au coeff (utilisateur1, utilisateur2) s'ils font partis de la même communauté.

Exemple : Si le coefficient (utilisateur1, utilisateur2) vaut 5, cela veut dire qu'ils appartiennent à la même communauté dans chaque sous graphe. Cela veut dire qu'ils ont des goûts similaires que ce soit sur les films qu'ils trouvent mauvais, moyens ou excellent.

On obtient donc la matrice suivante :



Cette matrice n’est rien d’autre que la matrice d’adjacence pondérée des nœuds utilisateurs.

On peut alors récupérer le graphe associé en utilisant la fonction *graph\_from\_adjacency\_matrix.*

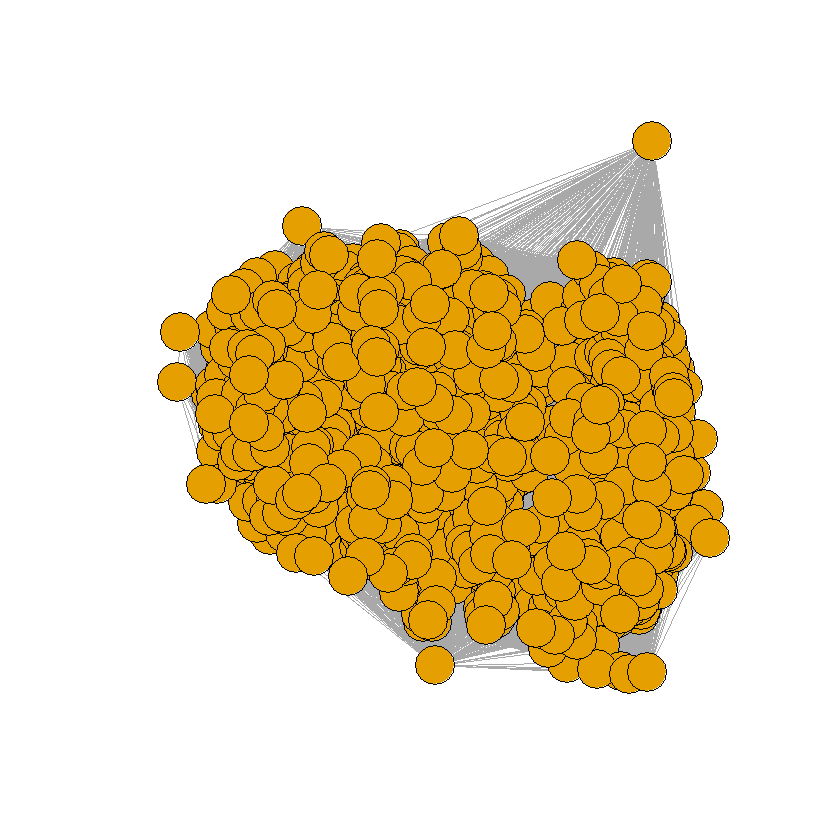
**

Figure - Graphe associé à la matrice d'adjacence représentant les liens entre les utilisateurs par rapport aux communautés détectés

Exemple : Sur ce graphe U2 et U3 auront un lien de poids 3, cela veut dire qu’ils se sont retrouvés 3 fois sur 5 dans la même communauté. Ce qui veut dire qu’ils ont des goûts similaires sur 3 notes de films sur 5.

1. Détection de communautés sur le nouveau graphe

En appliquant l’algorithme de Louvain sur ce nouveau graphe, on obtient le résultat suivant :

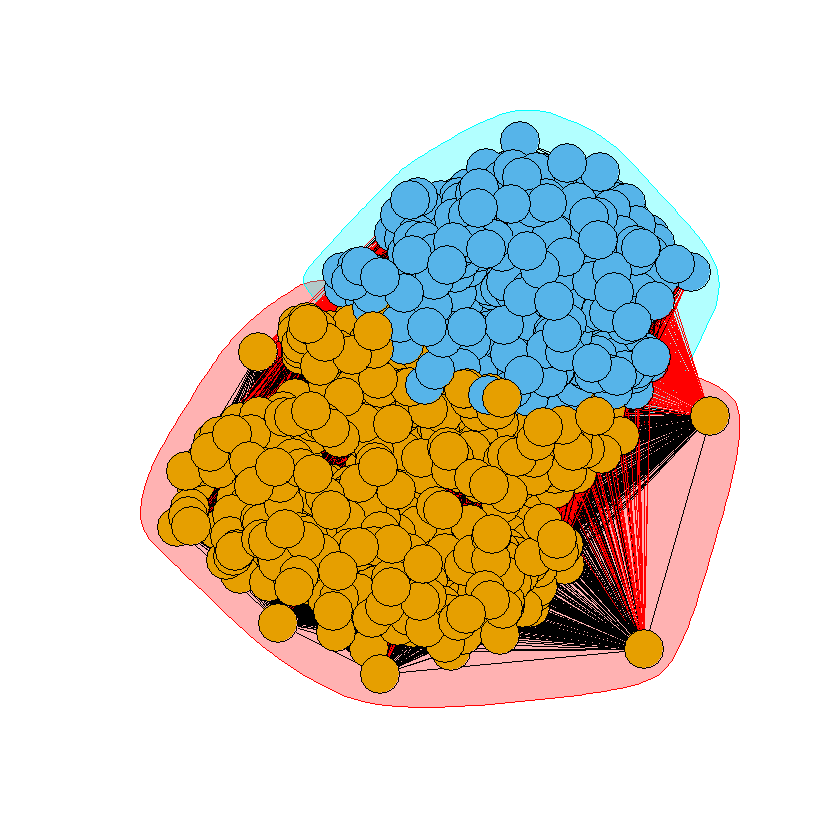


Figure - Communautés détectés par l'algorithme Louvain sur le graphe « global » des utilisateurs

1. Prédiction des liens

On a donc 2 grandes communautés d’utilisateurs. Pour remplir la matrice A de l’énoncé, nous allons utiliser la règle suivante :

Chaque lien manquant U­i-Ii au sein d’une communauté sera prédit comme étant la moyenne de toutes les notes accordées au film Ii par les autres utilisateurs de la communauté.

1. Complément : Utilisation de la deuxième projection

Jusqu’ici, nous avons utilisé uniquement la projection sur les utilisateurs. Or pour renforcer notre prédiction, on peut également faire la même chose avec la projection sur les utilisateurs.

On obtient la communauté suivante :

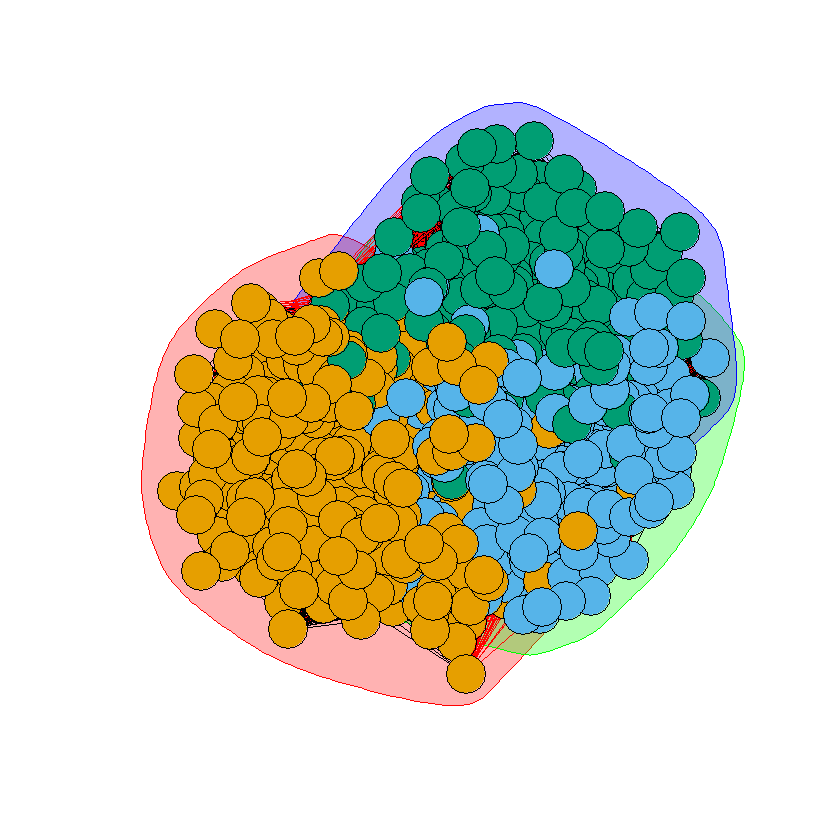


Figure - Communautés détectés par l'algorithme Louvain sur le graphe « global » des films

On obtient 3 grandes communautés de films, nous allons également compléter la matrice A à l’aide des ces communautés en suivant ces règles :

Chaque lien manquant U­i-Ii au sein d’une communauté sera prédit comme étant la moyenne de toutes les notes accordées par l’utilisateur Ui aux autres films de la communauté.

En cas de chevauchement avec les prédictions de la projection sur les utilisateurs, on prendra la moyenne des deux.

IV. Test

Grâce à cette méthode, nous sommes passés de 94% de lien manquant à 33%, maintenant il reste à évaluer la pertinence de cette méthode en testant la précision de ces prédictions.

Pour ce faire, nous allons supprimer 20% des liens dans le graphe initial et essayer de les prédire avec notre méthode.

Cela fait donc 20 000 à prédire, notre méthode ne permet pas de prédire la totalité des liens mais on a pu tout de même prédire 5039 liens sur les 20 000 supprimés :

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | MAE | Note exacte | 1 point d’écart sur la note réelle | 2 points d’écart sur la note réelle | 3 points d’écart sur la note réelle | 4 points d’écart sur la note réelle |
|  | 0.8 | 1802 (35%) | 2567 (49%) | 749 (14%) | 101 (2%) | 4 (0.07%) |

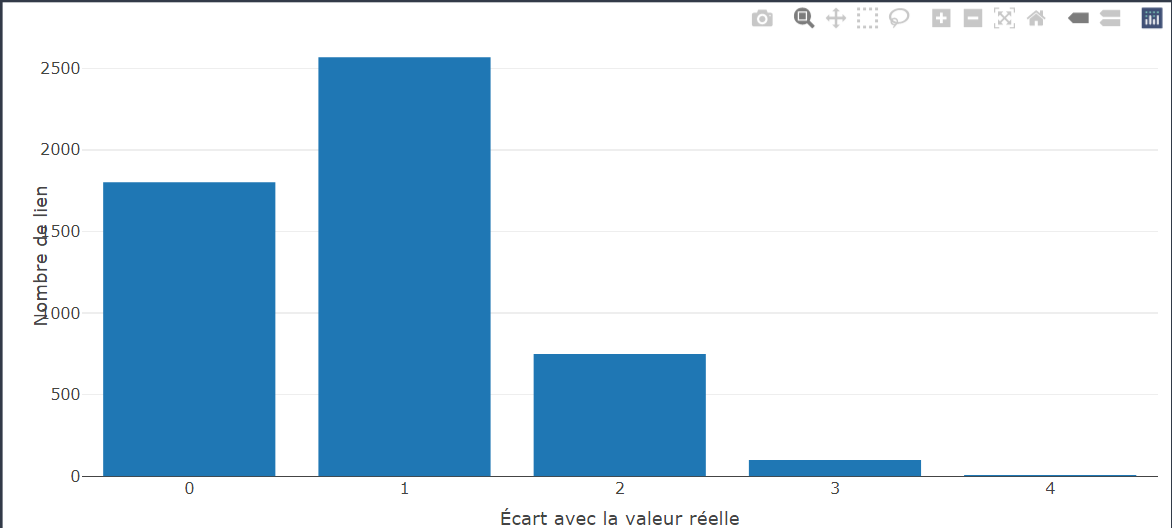


Figure - Histogramme représentant les écarts entre les valeurs prédites et les valeurs réelles des liens prédits

Dans la majorité des cas (85%), on se trompe maximum d’un point par rapport à la note réelle. Dans le cadre de notre problématique, un écart d’un point n’est pas dramatique. En effet, un film noté 4/5 et un film 5/5 renseigne autant sur l’appréciation de l’utilisateur, de même pour un film noté 1/5 et un film noté 2/5.

V. CONCLUSION

La solution proposée est pertinente dans le sens où les films que l’on va recommander sont cohérents d’après les résultats du test. Néanmoins, elle comporte quelques limites. Premièrement, cette méthode ne permet pas de prédire l’intégralité des liens manquants. En effet, dans les parties du graphe où l’on manque d’information (un film qui n’a été vu que par très peu de personnes ou un utilisateur qui ne regarde que des films très peu populaire par exemple), on ne peut rien faire avec cette méthode. On pourrait prédire plus de lien en étant moins exigeant sur les conditions d’appartenance à une communauté mais cela dégraderait la qualité des prédictions.

Deuxièmement, la détection de communauté détecte seulement 2 grandes catégories d'utilisateurs et seulement 3 catégories de films. La réalité est beaucoup plus nuancée que cela, en ce sens il serait peut-être intéressant de se pencher vers la détection de communautés locales pour les profils atypiques.

VI. Bibliographie

* <https://rstudio-pubs-static.s3.amazonaws.com/317838_9749880d51cb4946a096868c89e1f9b5.html>
* <https://medium.com/stanford-cs224w/graph-neural-network-based-movie-recommender-system-5876b9686df3>
* <https://freecontent.manning.com/creating-a-bipartite-graph-for-a-user-item-dataset/>
* <https://hal.archives-ouvertes.fr/hal-02474973/document>
* <https://www-complexnetworks.lip6.fr/~magnien/Publis/30Prediction/article.pdf>
* <https://www.frontiersin.org/articles/10.3389/fgene.2021.649440/full>