

Institut de Science Financière et d'Assurances (ISFA)

MASTER 2 - ECONOMETRIE, STATISTIQUES :
EQUADE

PROJET DATA MINING AVANCE - PARTIE 2

**Systèmes de recommandation de
films : Analyse de similarité et Réseaux
sociaux**

Etudiants :

CRÉSUS KOUNOUDJI
JOSUÉ NASSA
MADANY DIALLO

Professeure :
NASSIRA CHEKKAI

Mai 2023

Table des matières

1 Cadre général	2
1.1 Introduction	2
1.2 Objectifs du projet	3
2 Cadre Méthodologique	4
2.1 État de l'Art	4
2.2 L'analyse de similarité	4
2.2.1 Le modèle de prédiction	5
2.2.2 Évaluation des prédictions	6
2.3 Graphes et visualisation des réseaux sociaux	7
2.4 Le jeu de données et le dossier Python	8
3 Résultats	10
3.1 Partie 1	10
3.1.1 Les similarités	10
3.1.2 Les prédictions	11
3.1.3 Comparaison des évaluations et identification de la meilleure méthode	12
3.2 Partie 2	13
3.2.1 Densités de graphe des similarités <i>Cosinus, Jaccard et Pearson</i>	13
3.2.2 Analyses sur graphe de la matrice de similarité finale	14
3.3 Partie 3	19
3.3.1 Les réseaux sociaux	19
4 Conclusion et perspectives	21
Bibliographie	22

Chapitre 1

Cadre général

1.1 Introduction

Avec l'explosion de la quantité de contenus disponibles, les systèmes de recommandation jouent un rôle essentiel pour les plateformes de Vidéo à la demande (VOD) telles que *Netflix*, *Allocine* ou *Amazon Prime Video*, qui sont de facto les leaders dans ce domaine, afin d'aider les utilisateurs à découvrir de nouveaux films qui correspondent à leurs préférences et pouvoir les retenir plus longtemps [12].

L'analyse de similarité est une approche couramment utilisée dans les systèmes de recommandation de films [11]. Ces techniques permettent de comprendre les préférences des utilisateurs, d'identifier des modèles et des relations entre les utilisateurs et d'utiliser ces informations pour recommander des films qui correspondent à leurs goûts. Elles consistent à mesurer la similarité entre les utilisateurs sur la base de leurs évaluations des films. La similarité peut être calculée à l'aide de différentes mesures, notamment le coefficient de similarité *Cosinus*, le coefficient de *Jaccard* ou le coefficient de corrélation de *Pearson*. Ces mesures permettent de quantifier la proximité entre les utilisateurs ou les films, et servent de base pour recommander des films similaires à ceux appréciés par un utilisateur donné [1].

En outre, les graphes et les réseaux sociaux offrent une autre perspective pour comprendre les relations entre les utilisateurs dans un système de recommandation. Ce sont des outils puissants pour analyser les systèmes de recommandation [10]. En effet, en représentant les utilisateurs et les films comme des noeuds, et les interactions entre eux comme des arêtes, nous pouvons construire un graphe qui reflète les relations entre les entités. Les communautés au sein de ce graphe représentent des groupes d'utilisateurs ayant des intérêts similaires. En analysant les communautés de cette manière, nous pouvons mieux comprendre la structure du réseau social et identifier les utilisateurs principaux et les connecteurs, qui jouent un rôle clé dans la propagation de l'information entre les communautés.

Par ailleurs, des outils spécialisés tels que *Gephi* et *SocNetV* sont largement utilisés pour visualiser et explorer ces réseaux sociaux [6, 5]. *Gephi* est un logiciel open source qui offre des fonctionnalités avancées de visualisation et d'analyse des graphes et permet de représenter graphiquement les noeuds et les arêtes, d'appliquer des algorithmes de détection des communautés et de visualiser les résultats de manière interactive. *SocNetV* est un autre logiciel open source dédié à l'analyse des réseaux sociaux. Il fournit des outils pour la visualisation, l'analyse et la modélisation des réseaux sociaux, avec des fonctions telles que la détection des communautés, l'analyse de la centralité et la mesure des liens sociaux.

Cet projet se concentre sur les systèmes de recommandation de films basés sur l'analyse de

similarité et les réseaux sociaux. Dans ce contexte, nous utiliserons un ensemble de données [MovieLens Small Latest Dataset](#) qui contient les évaluations et les tags attribués par les utilisateurs aux films. A partir de [scripts Python](#), nous allons d'abord calculer les matrices de similarité entre les utilisateurs en utilisant différentes mesures (*Cosinus, Jaccard, Pearson*), puis utiliser ces matrices pour faire des prédictions et enfin créer un graphique représentant les relations entre les utilisateurs sur la base d'une similarité synthétique créée à partir des précédentes. À partir de ce graphe, nous identifierons les communautés et les utilisateurs principaux, et nous visualiserons enfin les graphes et les réseaux sociaux à l'aide des logiciels *Gephi* et *SocNetV*.

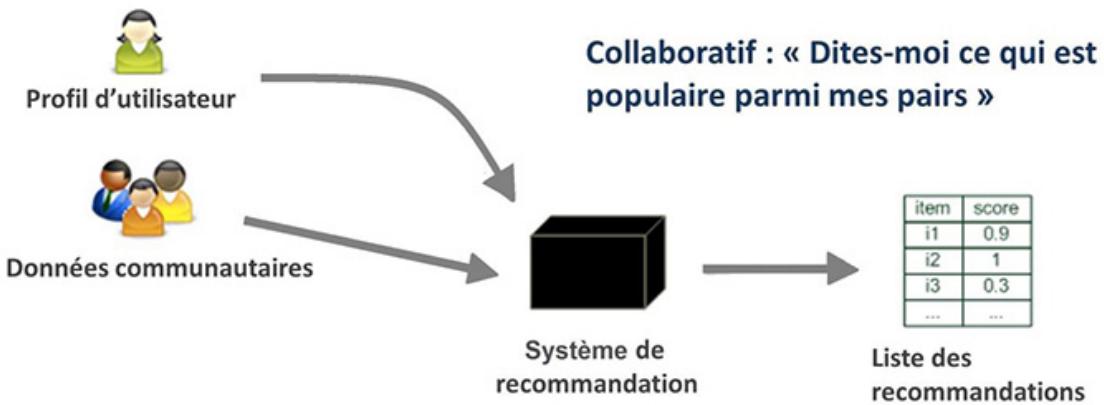


FIGURE 1.1 – Illustration d'un système de recommandation - approches basées sur le filtrage collaboratif [9].

1.2 Objectifs du projet

L'étude des systèmes de recommandation, en l'occurrence dans l'un des principaux champs d'application que sont les recommandations de films, est un domaine passionnant. Ce projet s'inscrit dans le cadre de la *Partie 2* du cours de *Data Mining Avancé* et vise à analyser les opinions des utilisateurs sur les films en utilisant l'ensemble de données [ratings.csv](#). L'objectif général est de réaliser les exercices préliminaires à la création d'un système de recommandation de films en utilisant des techniques de similarité entre utilisateurs basées sur les notations et l'exploration de graphes.

Chapitre 2

Cadre Méthodologique

2.1 État de l'Art

Dans la littérature scientifique, plusieurs travaux ont posé les bases des différentes approches dans le domaine de la recommandation de films, notamment en utilisant des analyses de similarité, de graphes et de réseaux sociaux. En particulier, plusieurs contributions présentent une approche de filtrage collaboratif basée sur les items, qui utilise la similarité entre les items pour recommander des films à travers différentes mesures de similarité, telles que le coefficient de similarité cosinus et le coefficient de corrélation de Pearson [11, 1]. Une autre approche notable est celle des techniques de factorisation matricielle pour les systèmes de recommandation, en l'occurrence l'algorithme de factorisation matricielle, qui décompose la matrice d'évaluation en deux matrices de facteurs latents pour représenter les utilisateurs et les films [7].

D'autres travaux ont souligné l'importance de l'analyse de graphes pour comprendre les interactions sociales et les structures communautaires dans les réseaux en comparant empiriquement plusieurs algorithmes de détection des communautés dans les réseaux sociaux et en évaluant les performances de différentes méthodes, telles que la méthode de Louvain et l'algorithme de propagation des étiquettes [8]. Des algorithmes tels que l'algorithme de détection des communautés [3] sont largement utilisés pour identifier les communautés dans les graphes des systèmes de recommandation.

Cette approche permet de regrouper les utilisateurs ayant des intérêts similaires et d'identifier les structures sociales au sein du système. Ces techniques de visualisation, d'analyse et de modélisation des graphes pour mesurer la proximité sociale, détecter les communautés et visualiser les réseaux sociaux sont faciles à mettre en œuvre sur des logiciels open source spécialisés tels que Gephi [2].

2.2 L'analyse de similarité

Dans ce projet, pour l'analyse de similarité des utilisateurs dans le contexte d'un système de recommandation de films, nous avons calculé différentes matrices de similarités à partir de trois coefficients de similarité : *Cosinus*, *Jaccard* et *Pearson*. Les matrices de similarité créées à partir de ces coefficients fournissent une mesure quantitative de la similarité entre les utilisateurs, ce qui permet d'identifier les utilisateurs ayant des goûts similaires et de recommander des films en fonction de ces similarités.

1. Coefficient de similarité *Cosinus*

Le coefficient de similarité *Cosinus* est efficace pour mesurer la similarité entre les utilisateurs lorsque la fréquence d'évaluation des films varie et est indépendante de l'échelle de notation des utilisateurs. Cependant, ce coefficient ne tient pas compte des évaluations manquantes, ce qui peut affecter la similarité lorsque les utilisateurs n'ont pas évalué les mêmes films. Le coefficient de similarité Cosinus est défini comme suit :

$$\text{Similarité Cosinus}(u_1, u_2) = \frac{\sum_{i=1}^n r_{1i} \cdot r_{2i}}{\sqrt{\sum_{i=1}^n r_{1i}^2} \cdot \sqrt{\sum_{i=1}^n r_{2i}^2}}$$

où u_1 et u_2 sont deux utilisateurs, r_{1i} et r_{2i} sont les évaluations des utilisateurs u_1 et u_2 pour le film i , et n est le nombre total de films.

2. Coefficient de similarité *Jaccard*

Le coefficient de similarité *Jaccard* est efficace pour mesurer la similarité entre les utilisateurs lorsque la fréquence d'évaluation des films est binaire (par exemple, aimé ou non aimé). De plus, il est indépendant de l'échelle de notation des utilisateurs. Toutefois, il ne tient pas compte des évaluations spécifiques des utilisateurs, ce qui peut conduire à une perte d'informations. Ce coefficient peut être défini par :

$$\text{Similarité Jaccard}(u_1, u_2) = \frac{|R_{u_1} \cap R_{u_2}|}{|R_{u_1} \cup R_{u_2}|}$$

où R_{u_1} et R_{u_2} sont les ensembles de films évalués par les utilisateurs u_1 et u_2 respectivement.

3. Coefficient de similarité *Pearson*

Quant au coefficient de similarité *Pearson*, il tient compte des différences dans les moyennes d'évaluation des utilisateurs et fonctionne bien même en présence d'évaluations manquantes. Mais il peut être sensible aux valeurs extrêmes et/ou affecté par les différences de variabilité entre les utilisateurs. On le définit par :

$$\text{Similarité Pearson}(u_1, u_2) = \frac{\sum_{i=1}^n (r_{1i} - \bar{r}_1) \cdot (r_{2i} - \bar{r}_2)}{\sqrt{\sum_{i=1}^n (r_{1i} - \bar{r}_1)^2} \cdot \sqrt{\sum_{i=1}^n (r_{2i} - \bar{r}_2)^2}}$$

où u_1 et u_2 sont deux utilisateurs, r_{1i} et r_{2i} sont les évaluations des utilisateurs u_1 et u_2 pour le film i , \bar{r}_1 et \bar{r}_2 sont les moyennes des évaluations des utilisateurs u_1 et u_2 , et n est le nombre total de films.

Une matrice de similarité finale est obtenue en prenant la moyenne des trois matrices de similarité. Elle représente une mesure globale de la similarité entre les utilisateurs.

Soit S_{Cosinus} , S_{Jaccard} et S_{Pearson} les matrices de similarité basées respectivement sur les coefficients de similarité *Cosinus*, *Jaccard* et *Pearson*. La matrice de similarité finale S_{Final} est définie comme suit :

$$S_{\text{Final}} = \frac{S_{\text{Cosinus}} + S_{\text{Jaccard}} + S_{\text{Pearson}}}{3}$$

2.2.1 Le modèle de prédiction

Un modèle de prédiction a été défini pour prédire les valeurs de notes à partir de matrice de similarité. Ainsi, nous utilisons la formule de prédiction suivante pour réaliser des prédictions à partir de matrices de similarité conformément aux consignes :

$$\text{prediction}(\text{user}, \text{item}) = \frac{\sum_{\text{otheruser}} \text{similarity}(\text{user}, \text{otheruser}) \times \text{rating}(\text{otheruser}, \text{item})}{\sum_{\text{otheruser}} \text{similarity}(\text{user}, \text{otheruser})}$$

où $\text{prediction}(\text{user}, \text{item})$ est la valeur prédictée de l'évaluation de l'utilisateur pour un film donné, $\text{similarity}(\text{user}, \text{otheruser})$ est la similarité entre l'utilisateur et un autre utilisateur, et $\text{rating}(\text{otheruser}, \text{item})$ est l'évaluation de l'autre utilisateur pour le film.

Cette formule calcule une moyenne pondérée des évaluations des autres utilisateurs, en utilisant la similarité comme poids. Ainsi, les utilisateurs similaires auront un poids plus élevé dans la prédiction. Il convient de noter que dans notre script, si la similarité totale ($\sum_{\text{otheruser}} \text{similarity}(\text{user}, \text{otheruser})$) est égale à zéro, la prédiction sera également zéro pour éviter une division par zéro.

2.2.2 Évaluation des prédictions

Les évaluations des prédictions sont essentielles pour mesurer la performance des systèmes de recommandation. Deux mesures couramment utilisées sont le *rappel (recall)* et l'*erreur absolue moyenne (MAE)*.

1. Recall

Le *rappel (Recall)* est une mesure qui évalue la capacité du système de recommandation à identifier les éléments pertinents dans le top-k ($k = 5$) de la liste de classement. Il est calculé en divisant le nombre d'éléments pertinents inclus dans le top-k par le nombre total d'éléments pertinents dans l'ensemble de test. La formule est utilisée pour le calcul du rappel :

$$\text{recall} = \frac{\sum_u N(k, u)}{\sum_u N(u)}$$

où $N(k, u)$ est le nombre d'éléments pertinents pour l'utilisateur u dans le top-k et $N(u)$ est le nombre total d'éléments pertinents pour l'utilisateur u dans l'ensemble de test.

2. MAE

L'*erreur absolue moyenne (MAE)* est une mesure qui évalue la différence moyenne absolue entre les évaluations réelles et les prédictions. Elle est calculée en prenant la somme des valeurs absolues des différences entre les prédictions et les évaluations réelles, puis en divisant par le nombre total d'évaluations. La formule suivante représente le calcul du *MAE* :

$$\text{MAE} = \frac{\sum_{i,j} |P_{i,j} - r_{i,j}|}{n}$$

où $P_{i,j}$ est la prédiction pour l'élément i de l'utilisateur j , $r_{i,j}$ est l'évaluation réelle pour l'élément i de l'utilisateur j , et n est le nombre total d'évaluations.

Ainsi, le rappel est-il une mesure intuitive qui évalue la capacité à recommander des éléments pertinents. Il est particulièrement utile dans les systèmes de recommandation où la pertinence des éléments est cruciale, comme les recommandations de films. Cependant, il ne tient pas compte de la précision des prédictions, ce qui peut être un inconvénient si l'on souhaite privilégier des recommandations précises plutôt que de simplement recommander des éléments pertinents. L'*erreur absolue moyenne (MAE)*, quant à elle, évalue la précision des prédictions en calculant la différence absolue moyenne entre les prédictions et les évaluations réelles. C'est

une mesure largement utilisée dans les systèmes de recommandation car elle permet de quantifier la performance globale du modèle. Cependant, elle ne prend pas en compte la pertinence des prédictions, ce qui peut être un inconvénient si l'on souhaite privilégier des recommandations pertinentes plutôt que de simplement minimiser l'erreur. Ainsi, il est important d'utiliser de manière complémentaire ces méthodes pour évaluer la performance d'un système de recommandation. En combinant le *rappel* et le *MAE*, on peut obtenir une vision plus complète de la performance en tenant compte à la fois de la pertinence des recommandations et de la précision des prédictions.

2.3 Graphes et visualisation des réseaux sociaux

Pour la représentation et l'analyse des réseaux sociaux, on utilise des techniques basées sur les graphes. Dans cette section, nous avons d'abord généré un graphe à partir des matrices de similarité afin de calculer les densités des graphes pour les comparer. Ensuite, nous utilisons la matrice de similarité finale entre les utilisateurs pour identifier les *utilisateurs leaders* et *utilisateurs connecteurs*, ainsi que les *meilleures relations permettant la circulation d'informations entre les communautés*. Enfin, pour explorer les réseaux sociaux de la similarité sous forme de graphe, nous avons exporté cette dernière au format adéquat (*.csv*) pour l'importer dans les logiciels *Gephi* et *SocNetV*.

Ainsi, grâce à la fonction `from_numpy_array()` du module `networkx` de Python, une matrice de similarité est convertie en un graphe non orienté, où les utilisateurs sont représentés par des noeuds et les relations entre les utilisateurs sont représentées par des arêtes.

1. Densité de graphe

La densité d'un graphe mesure le degré de connectivité entre les utilisateurs et permet de comparer différentes mesures de similarité. Une densité élevée indique une forte interconnexion entre les utilisateurs, tandis qu'une densité faible indique une faible interconnexion. L'avantage de cette méthode est qu'elle permet de représenter les relations entre les utilisateurs de manière visuelle et d'analyser la structure globale du réseau social. Cependant, elle ne tient pas compte des relations spécifiques entre les utilisateurs et peut ne pas refléter complètement la dynamique du réseau social. De plus, elle peut être limitée par la taille de la matrice de similarité et la complexité du calcul de la densité pour de grands réseaux sociaux.

La formule pour calculer la densité d'un graphe à partir d'une matrice de similarité est donnée par :

$$\text{Densité du graphe} = \frac{\text{Nombre d'arêtes dans le graphe}}{\text{Nombre maximal d'arêtes possible}}$$

Les densités sont calculées ici à partir de la commande `networkx.graph.density()` appliquée aux objets (graphes non orientés) générés par la fonction `from_numpy_array()` du module `networkx`.

2. 5 Utilisateurs leaders

Les utilisateurs leaders sont des utilisateurs influents dans un réseau social qui ont un grand nombre de voisins. Ils jouent un rôle central dans la diffusion d'informations et l'influence des autres utilisateurs. Il sont obtenus en triant les utilisateurs par ordre

décroissant des degrés donnés par la commande `graph.degree()` appliquée au graphe non orienté généré à partir de la matrice de similarité finale à l'aide de la fonction `from_numpy_array()`.

3. 5 Utilisateurs connecteurs

Les utilisateurs connecteurs sont ceux qui relient le maximum de communautés dans le graphe. Pour identifier les 5 utilisateurs connecteurs, nous utilisons la mesure de centralité intermédiaire (betweenness centrality) pour calculer l'importance des utilisateurs dans la communication entre les communautés. Les utilisateurs connecteurs sont triés par leur coefficient de centralité intermédiaire décroissant calculés à l'aide de la fonction `betweenness_centrality()` du module `networkx` de Python.

4. 5 Meilleures relations d'information :

Les meilleures relations d'information sont les arêtes du graphe par lesquelles les informations circulent le plus d'une communauté à une autre. Pour identifier les 5 meilleures relations d'information, nous utilisons la fonction `edge_betweenness_centrality()` du module `networkx` de Python. Cette fonction utilise la mesure de centralité intermédiaire.

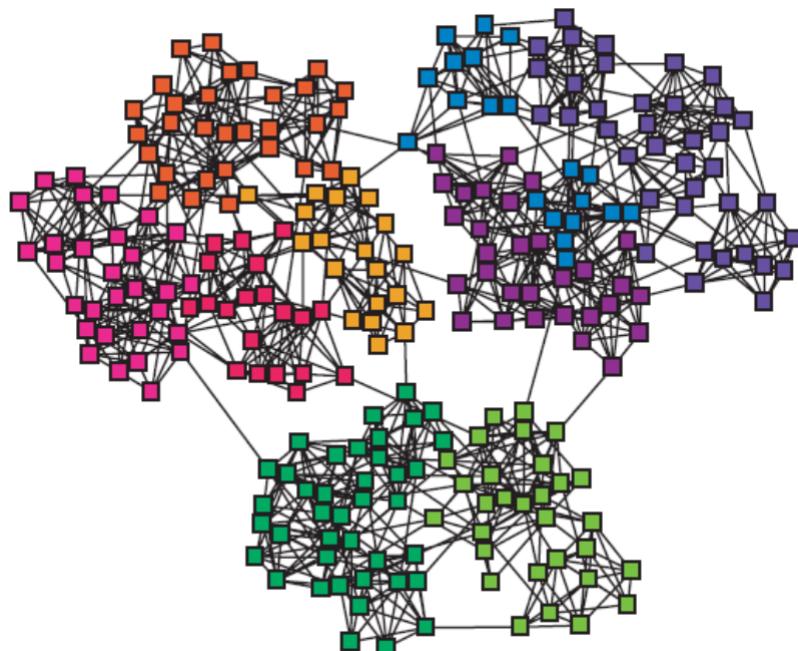


FIGURE 2.1 – Illustration fouille de graphes et réseaux sociaux [4].

2.4 Le jeu de données et le dossier Python

Les données [Movie Lens Small Latest Dataset](#) utilisées pour ce projet sont issues de la plateforme Kaggle et décrivent l'activité d'évaluation à 5 étoiles et de marquage en texte libre de *MovieLens*, un service de recommandation de films. Plus spécifiquement, nous avons utilisé

la table de données *ratings.csv* qui contient 100 836 évaluations et 3 683 applications de tags sur 9 742 films. Ces données ont été créées par 610 utilisateurs entre le 29 mars 1996 et le 24 septembre 2018. Cet ensemble de données a été généré le 26 septembre 2018. Chaque ligne de ce fichier représente une évaluation (colonne : *Rating*) sur une échelle de 5 étoiles, avec des incrémentés d'une demi-étoile (0.5 étoile - 5.0 étoiles) d'un film (colonne : *movieId*) par un utilisateur (colonne : *userId*) et avec une colonne de dates (colonne : *Timestamps*) qui représente les secondes écoulées depuis minuit en temps universel coordonné (UTC) le 1er janvier 1970. Les utilisateurs ont été sélectionnés au hasard. Tous les utilisateurs sélectionnés ont évalué au moins 20 films. Aucune information démographique n'est incluse. Chaque utilisateur est représenté par un identifiant et aucune autre information n'est fournie.

Movie Lens Small Latest Dataset

Data Card Code (50) Discussion (0)

53

New Notebook

Download (994 kB)



ratings.csv (2.48 MB)

↓ □ >

Detail Compact Column

4 of 4 columns

userId	movieId	rating	timestamp
1	1	4.0	964982703
1	3	4.0	964981247
1	6	4.0	964982224
1	47	5.0	964983815
1	50	5.0	964982931
1	70	3.0	964982400
1	101	5.0	964980868
1	110	4.0	964982176
1	151	5.0	964984041
1	157	5.0	964984100
1	163	5.0	964983650
1	216	5.0	964981208
1	223	3.0	964980985
1	231	5.0	964981179
1	235	4.0	964980908
1	260	5.0	964981680

Data Explorer

Version 1 (3.3 MB)

- README.txt
- links.csv
- movies.csv
- ratings.csv**
- tags.csv

Summary

FIGURE 2.2 – Aperçu de la table de données *ratings.csv* du Kaggle - Movie Lens Small Latest Dataset.

Les analyses ont été effectuées à partir de scripts exécutés sur Python 3.11. D'abord écrit dans un Jupyter Notebook le code est répartis en modules de fonctions et un fichier principal (*.py*) pour créer un dossier Python. Le dossier Python, entre autres, est disponible sur le dépôt GitHub [projet-datamining-recommandation-de-films-](#) dédié.

Chapitre 3

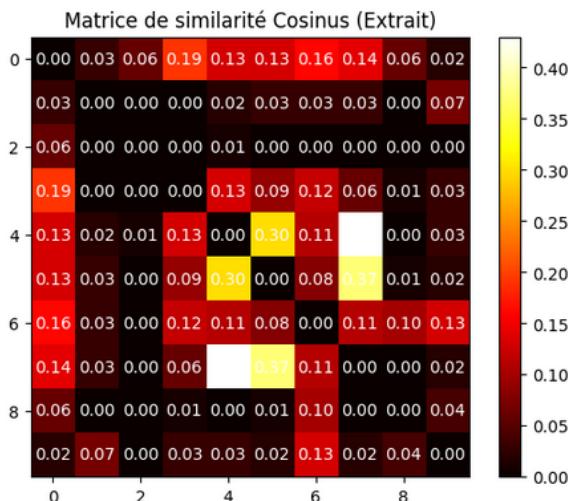
Résultats

3.1 Partie 1

3.1.1 Les similarités

Il s'agit dans cette première partie, de procéder à une analyse de similarité à partir des matrices de similarités construites à partir de trois coefficients de similarité selon les équations 1, 3 et 2 qui calculent les similarités user-user. Tous les résultats de cette partie 1 sont issues du script Python. Les figures 3.1, 3.3 et 3.2 ci-dessous donnent un aperçu graphique d'un petit échantillon (dix (10) premières lignes et colonnes) de ces différentes matrices de similarités que nous avons calculer dans un premier temps. Chaque matrice a une dimension de six cent dix lignes (610) et colonnes, qui correspondent au nombre d'utilisateurs uniques dans la base identifiés par la colonne *userId*. Il est important de noter que dans notre code de calcul, nous avons choisi d'annuler la diagonale principale des matrices de similarité pour éviter la similarité avec soi-même. Cela est généralement fait pour les systèmes de recommandation afin d'éviter que l'élément lui-même ne soit considéré comme similaire à lui-même.

Matrice de similarité Cosinus : Dimension = (610, 610)



Matrice de similarité Pearson : Dimension = (610, 610)

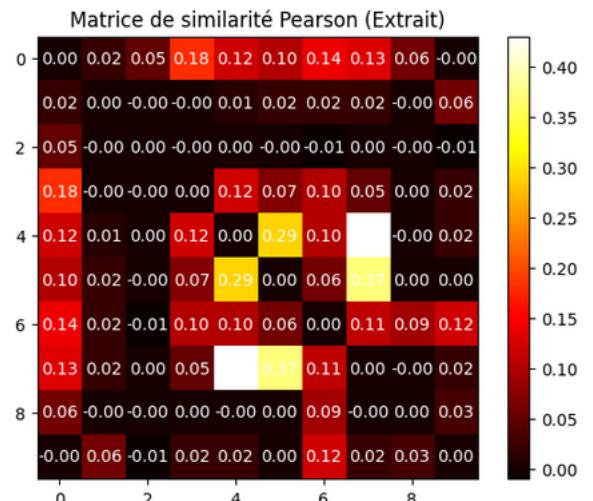


FIGURE 3.1 – Extrait de la matrice de similarités *Cosinus*

FIGURE 3.2 – Extrait de la matrice de similarités *Pearson*

Matrice de similarité Jaccard : Dimension = (610, 610)

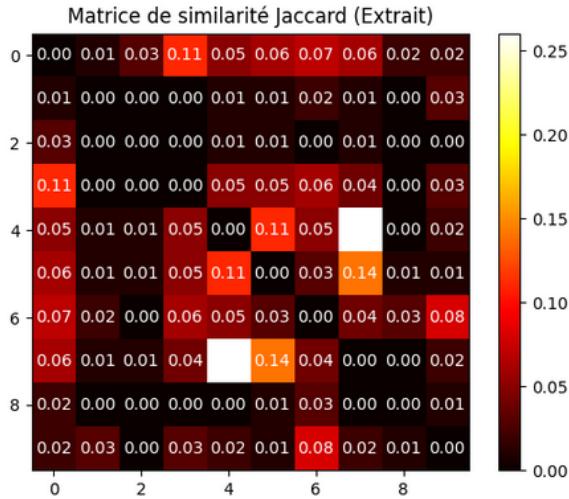


FIGURE 3.3 – Extrait de la matrice de similarités *Jaccard*

Dans un second temps, nous avons construit une matrice de similarité finale qui est la moyenne des trois précédentes comme défini plus haut dans l'équation 2.2. Cette matrice a été normalisé pour faire varier ses valeurs en 1 et 2. Toutes les matrices ont ensuite été sauvegardé au format *.txt*. La figure 3.4 donne une représentation graphique d'un petit échantillon (dix (10) premières lignes et colonnes) de la matrice de similarité finale ainsi construite.

Matrice de similarité finale : Dimension = (610, 610)

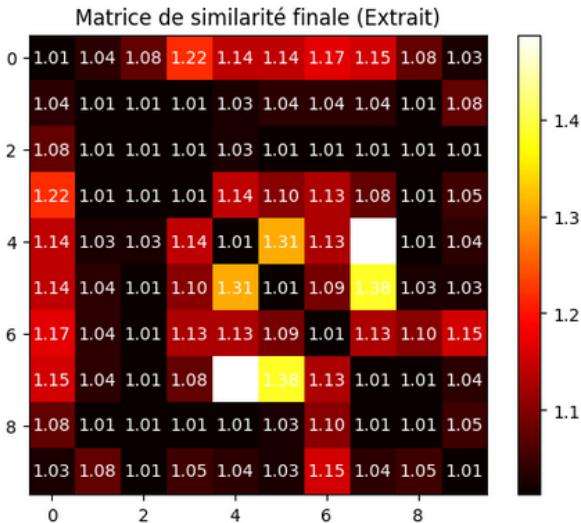


FIGURE 3.4 – Extrait de la matrice de similarités finale

3.1.2 Les prédictions

Une fois les matrices de similarité construites à l'aide de notre script, nous avons sélectionner aléatoirement cinq (05) vote que nous avons supprimer. Ainsi, à partir de la fonction de prédiction définit avec l'équation 2.2.1, nous avons utilisés les valeurs restantes auxquelles nous avons appliqués la matrice de similarité finale conformément à l'équation afin de de prédire les valeurs manquantes. Les tableaux 3.5 et 3.6 présentent respectivement les valeurs supprimées et les prédictions correspondantes réalisées à partir de la matrice de similarité et des données restantes.

Votes sélectionnés à supprimer pour prédictions : Valeurs de prédiction pour les votes supprimés :

	userId	movieId	rating	timestamp
67037	432	77866	4.5	1335139641
42175	288	474	3.0	978465565
93850	599	4351	3.0	1498524542
6187	42	2987	4.0	996262677
12229	75	1610	4.0	1158989841

FIGURE 3.5 – Votes supprimés aléatoirement.

	userId	movieId	rating_predicted	timestamp
67037	432	77866	2.914634	1335139641
42175	288	474	3.654311	978465565
93850	599	4351	3.292647	1498524542
6187	42	2987	3.590139	996262677
12229	75	1610	3.934426	1158989841

FIGURE 3.6 – Valeurs de prédites.

3.1.3 Comparaison des évaluations et identification de la meilleure méthode

Dans cette sous-section, on s'intéresse à la performance des trois matrices de similarité les une par rapport aux autres, dans la fonction de prédiction. Pour cela, nous allons comparer les évaluations des trois similarités (*Cosine*, *Jaccard*, *Pearson*) en termes de *rappel* et de *MAE*. Nous avons commencé par sélectionné un échantillon aléatoire de dix (10) votes à prédire pour l'évaluation que nous avons donc supprimé dans un premier temps. Ensuite, nous avons passé la table restante dans la fonction de prédiction que nous avons itéré trois fois en remplaçant à chaque itération la matrice de similarité par l'une des trois matrices (*Cosine*, *Jaccard*, *Pearson*). Il s'agit enfin de calculer d'une le *MAE* comme décrit dans l'équation 2 et d'autre par le *Rappel* en mesurant la fraction des éléments pertinents dans l'ensemble de test qui sont inclus dans le top-k de la liste de classement avec $k = 5$ (équation 1). Après avoir effectué ces calculs, nous avons identifié la meilleure méthode en comparant les valeurs de *MAE* et de *rappel*. La méthode avec le plus faible *MAE* est considérée comme la meilleure en termes de précision des prédictions, tandis que la méthode avec le plus élevé *Rappel* est considérée comme la meilleure en termes de pertinence des recommandations (ici, le *Rappel* est commun aux trois méthodes).

Comme le montre le résultats 3.7 ci-après, la méthode *Jaccard* a été identifiée comme la meilleure méthode dans notre cas. En effet, la prédiction par la matrice de similarité *Jaccard* a obtenu le plus faible *MAE* (0.58) parmi les trois méthodes, ainsi qu'un *rappel* de 0.6, qui est équivalent à celui des autres méthodes.

```
Meilleure méthode : Jaccard
MAE : 0.58
Recall : 0.6
```

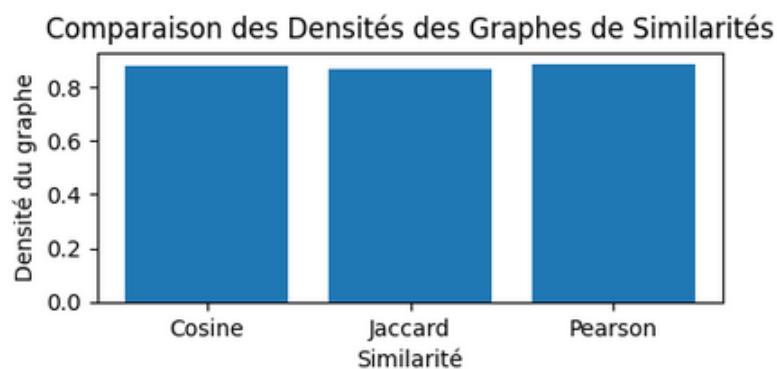
	Method	MAE	Recall
0	Cosine	0.61	0.6
1	Jaccard	0.58	0.6
2	Pearson	0.60	0.6

FIGURE 3.7 – Évaluation des trois similarités en termes de rappel (recall) et MAE.

3.2 Partie 2

3.2.1 Densités de graphe des similarités *Cosinus*, *Jaccard* et *Pearson*

L'analyse de la densité d'un graphe donne des informations sur le degré de connectivité entre les utilisateurs. Il apparaît donc pertinent de comparer les graphes générés à partir des différentes matrices de similarité pour les comparer du point de vu des interconnexions entre les utilisateurs. Nous avons à cet effet créé des graphes non orientés à partir de chaque matrice dans script Python. La figure 3.8 montre les valeurs des densités calculés et tandis que la figure 3.9 donne un aperçu des graphes en question pour mettre en perspectives les valeurs de densité calculé. Cette seule dernière ne tenant pas compte des relations spécifiques entre les utilisateurs et ne pouvant refléter complètement la dynamique du réseau social.



Le graphe de similarité 'Pearson' a la plus grande densité de graphe.
Tableau de valeurs des densités calculées :

	Similarité	Densité du graphe
0	Cosine	0.877596
1	Jaccard	0.866317
2	Pearson	0.885741

FIGURE 3.8 – comparaison de la densité des graphes des trois matrices de similarité.

Aperçu des graphes des matrices de similarité :

Le graphe de similarité 'Pearson' a la plus grande densité de graphe.

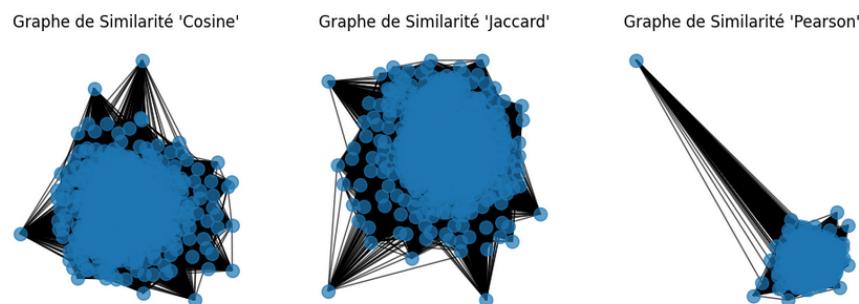


FIGURE 3.9 – Aperçu des graphes des matrices de similarité pour mise en perspective avec les calculs de densités.

L'analyse des résultats ci-dessus ne permettent pas de conclure puisque même si la valeurs de la densité de la similarité *Pearson* est la plus grande, les différences sont infimes.

3.2.2 Analyses sur graphe de la matrice de similarité finale

Dans cette sous section de cette deuxième partie, pour finir de jeter les bases du système de recommandation de film, on souhaite grâce aux analyses basées sur les graphes, procéder à l'identification de caractéristiques clés du graphe de la similarité finale. Nous avons généré un graphe non orienté à partir de cette dernière à l'aide de notre script Python. La figure 3.10 donne un aperçu de ce graphe (avec une détection de communauté à l'aide de la fonction `best_partition()` du module `community_louvain` de Python pour rendre la figure plus esthétique).

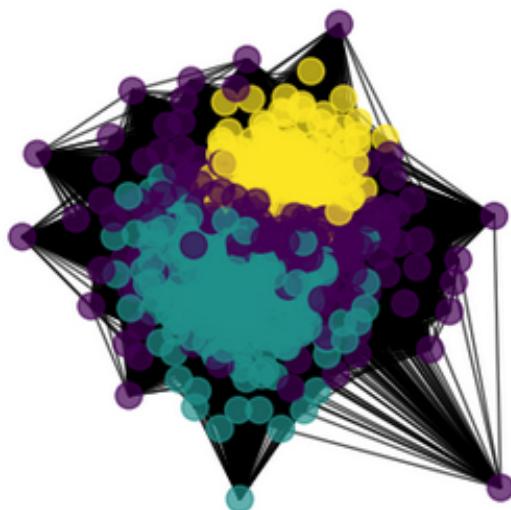


FIGURE 3.10 – Visualisation du graphe non orienté généré à partir de la matrice finale de similarité (avec détection des communautés)

A partir de ce graphe non orienté et à l'aide de fonction personnalisée écrit sur Python nous avons identifié :

1. les 5 utilisateurs leaders ayant le maximum de voisins

Sous-graphe des Utilisateurs Leaders (609 voisins)

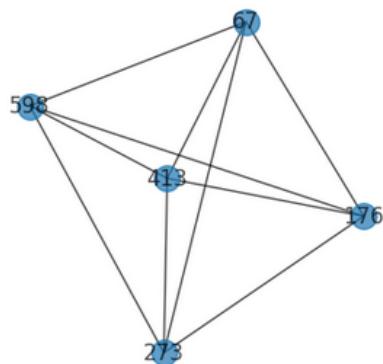


FIGURE 3.11 – Aperçu du sous-graphes des 5 utilisateurs leaders.

Nous avons choisi de donner une représentation graphique [3.11] des utilisateurs clés identifiés par notre script. On distingue dans la table de données associée les utilisateurs 67 ; 598 ; 413 ; 176 et 273 comme utilisateurs leaders avec 609 voisins pour les quatre (04) premiers et respectivement 607 et 608 voisins pour les deux (02) restants.

2. les 5 utilisateurs connecteurs qui relient le maximum de communautés

Dans cette autre représentation graphique ci-dessous [3.12], les utilisateurs connecteurs identifiés, sont les utilisateurs 67 ; 598 ; 413 ; 176 et 605 avec un coefficient de centralité commun de 4.78×10^{-4} . On retrouve tous les utilisateurs leaders sauf l'utilisateur 273 qui cède sa place ici dans le top cinq (05) à l'utilisateur 605.

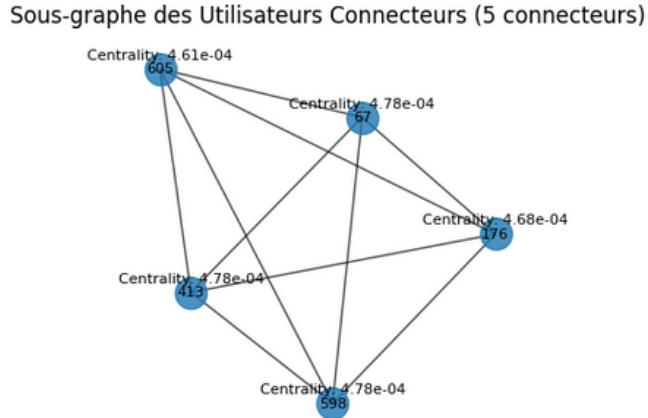


FIGURE 3.12 – Aperçu du sous-graphes des 5 utilisateurs Connecteurs.

3. 5 meilleures relations par lesquelles les informations circulent d'une communauté à une autre

Pour ce qui concerne les meilleures relations, on distingue les relations formées par les paires de nœuds (174, 598) ; (67, 174) ; (174, 413) ; (174, 479) et (174, 176) avec l'utilisateur 174 au carrefour des communautés les autres utilisateurs étant à une exception près les utilisateurs leaders et connecteurs. De plus la représentation graphique 3.13 qu'on propose montre que cinq (05) communautés différents sont représentées dans ces relations. Le coefficient de centralité est commun et est de 5×10^{-5}

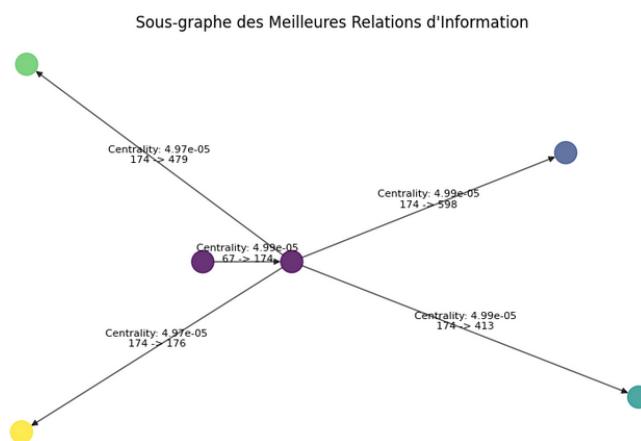


FIGURE 3.13 – Aperçu du sous-graphes des 5 meilleures relations avec coloration des nœuds selon la communauté reliée.

Après l'identification des Utilisateurs clés dans le graphe, l'étape suivante vers le système de recommandation de film est l'identification d'ensembles clés de film. Ainsi :

- 1. Identification du plus petit ensemble de films qui permettent d'atteindre tous les genres et styles que l'utilisateur a déjà regardés**

Pour identifier le plus petit ensemble de films qui permettent d'atteindre tous les genres et les styles que l'utilisateur a déjà regardés, un graphe bipartite pondéré (avec couverture minimum) est un type de graphe adéquat. En effet, un graphe bipartite, les noeuds sont divisés en deux ensembles distincts, et il n'y a pas d'arêtes entre les noeuds d'un même ensemble. Les arêtes du graphe représentent la relation entre les films et les genres et/ou les styles. Chaque arête est pondérée pour indiquer à quel point un film est associé à un genre ou style donné. Ainsi ici, un ensemble représentera les films et l'autre ensemble représentera les genres ou les styles et on pourra pondérer les arêtes en nous basant sur des mesures telles que le nombre de fois qu'un genre ou un style apparaît dans les films ou une mesure de similarité entre le film et le genre ou style. De fait, le paramètre de graphe que nous utiliserons est le poids des arêtes. En attribuant les bons poids aux arêtes, on pourra représenter la pertinence de chaque film dans chaque genre ou style. Ce qui permettra de trouver le plus petit ensemble de films qui couvrent tous les genres et les styles que l'utilisateur a déjà regardés. Une fois le graphe bipartite pondéré construit, on pourra utiliser des algorithmes de couverture de sommets ou de couverture de sommets pondérés pour trouver le plus petit ensemble de films qui couvre tous les genres et les styles. Avec ces algorithmes, on cherche à sélectionner un ensemble de noeuds de manière à ce que chaque genre ou style soit représenté une fois au moins. Ce type de graphe permettra de résoudre efficacement le problème d'identification et de recommander des films qui correspondent aux préférences de l'utilisateur tout en couvrant tous les genres et les styles qu'il a déjà explorés.

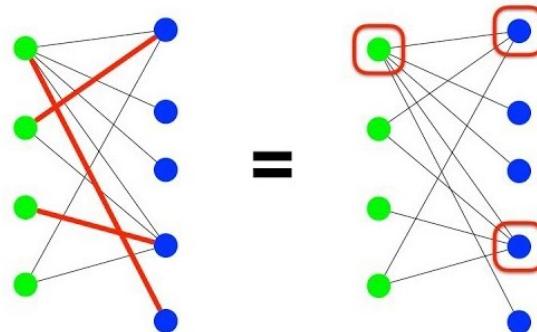


FIGURE 3.14 – illustration du théorème de Konig (couverture dans les graphes bipartis)

- 2. Identification de l'ensemble de films qui n'ont jamais été recommandés ensemble**

Pour identifier l'ensemble de films qui n'ont jamais été recommandés ensemble, une solution serait d'utiliser un graphe non orienté. Ce type de graphe pourrait être approprié puisqu'il permettrait de modéliser les relations entre les films et de déterminer les films qui ne sont jamais recommandés simultanément. Dans ce graphe, chaque noeud correspondra à un film, et les arêtes connecteront les paires de films qui ont été recommandés ensemble au moins une fois. Si deux films ont été recommandés ensemble, une arête est

ajoutée entre les deux noeuds correspondants. La construction de ce graphe implique donc de parcourir les données de recommandation et de créer les arêtes correspondantes entre les paires de films recommandés ensemble. Étant donné que nous nous intéressons uniquement à la présence ou à l'absence de recommandation simultanée entre les films, les arêtes seront non pondérées. Une fois le graphe de non orienté construit, on pourra utiliser des algorithmes de recherche de cliques, tels que l'algorithme de recherche de cliques maximales (l'algorithme de *Bron-Kerbosch* par exemple), pour identifier les ensembles de noeuds (films) qui forment des cliques complètes, c'est-à-dire des ensembles de films qui ne sont jamais recommandés ensemble. On peut définir une clique comme étant un sous-graphe où tous les noeuds sont directement connectés les uns aux autres. Les cliques correspondront donc à ce qu'on cherche à identifier. L'utilisation combinée de graphe non orienté et des algorithmes de recherche de cliques permet une identification efficace de l'ensemble de films qui n'ont jamais été recommandés ensemble. Une alternative aux algorithmes de recherche de cliques consiste à recourir à des algorithmes de recherche de composantes connexes dans le graphe, tels que l'algorithme de parcours en profondeur (*DFS*) ou l'algorithme de parcours en largeur (*BFS*). Un algorithme de parcours en profondeur permet d'explorer le graphe en profondeur en commençant par un noeud initial, et il permet de trouver toutes les composantes connexes du graphe. Ici, les composantes connexes représentent les ensembles de films qui ont été recommandés ensemble. Une composante connexe étant un sous-graphe où tous les noeuds sont accessibles les uns des autres par des chemins dans le graphe. En utilisant l'algorithme de parcours en profondeur sur le graphe des recommandations, nous pouvons identifier toutes les composantes connexes, puis sélectionner l'ensemble de films qui n'appartiennent à aucune composante connexe. Ces films sont ceux qui n'ont jamais été recommandés ensemble. L'avantage de cette approche est qu'elle nous permet d'identifier directement les films qui n'ont jamais été recommandés ensemble, en utilisant les informations sur les recommandations passées. Cela peut être utile pour diverses applications, telles que la diversification des recommandations en présentant de nouveaux films aux utilisateurs.

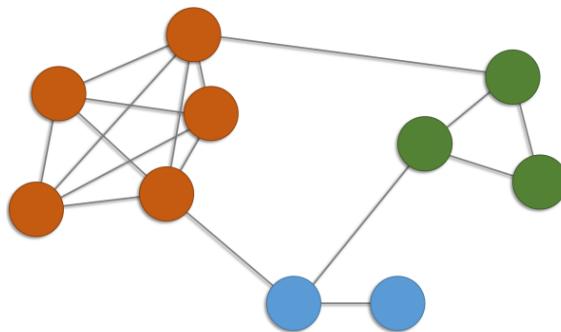


FIGURE 3.15 – Illustration Nœuds codés par couleur en fonction de l'appartenance à une clique.

3. Recommandation d'un film à un utilisateur en s'assurant que les films recommandés appartiennent à différents types

Pour recommander des films à un utilisateur en veillant à ce que les films recommandés appartiennent à différents types, un graphe biparti pondéré de similarité des types de films serait approprié. En effet, ce type de graphe permet de modéliser les relations entre les films et leurs types, et de prendre en compte la similarité entre les types de films lors

de la recommandation. Dans ce graphe biparti, on aurait deux ensembles de nœuds : les nœuds du premier ensemble représentent les films, tandis que les nœuds du deuxième ensemble représentent les types de films. Les arêtes pondérées reliant les films aux types correspondants en fonction de la similarité entre les types de films. On pourra se baser sur la similarité de *Jaccard* entre les types de films (meilleure ici à priori) pour cette pondération. La construction de ce graphe implique de parcourir les données de films et de types de films, de calculer la similarité entre les types de films et d'ajouter les arêtes pondérées correspondantes entre les films et les types. Plus la similarité entre les types de films est élevée, plus le poids de l'arête est élevé. Après construction de ce graphe biparti des types de films, on pourra utiliser un algorithme de recommandation basé sur la diffusion d'informations, pour recommander des films à un utilisateur ; l'algorithme Influence maximale (Maximal Influence) par exemple. L'algorithme Influence maximale cherche à maximiser l'influence d'une recommandation en identifiant les utilisateurs clés à partir desquels l'information peut se propager efficacement dans le réseau. Il se base sur des mesures d'influence et d'éloignement dans le réseau pour déterminer les utilisateurs à cibler et les éléments à recommander. Cet type d'algorithme prend en compte la similarité des types de films lors de la diffusion des recommandations et assure que les films recommandés appartiennent à différents types. De plus, l'utilisation d'un graphe biparti pondéré avec des algorithmes de recommandation basés sur la diffusion d'informations garanti que les films recommandés à un utilisateur appartiennent à différents types. Ce qui permet de diversifier les recommandations et d'offrir à l'utilisateur une expérience plus variée en explorant différents genres et styles de films. Une alternative aux algorithmes de recommandation basés sur la diffusion d'informations consisterait à utiliser des algorithmes de recherche de plus courts chemins dans le graphe, tels que l'algorithme de *Dijkstra*, pour trouver les films appartenant à différents genres avec les pondérations les plus élevées.

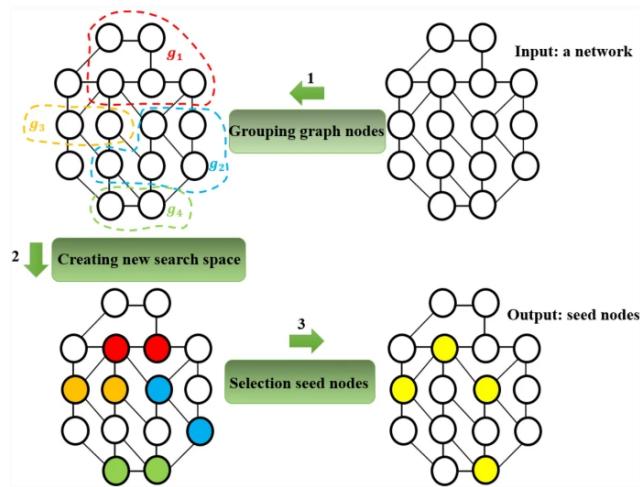


FIGURE 3.16 – Illustration d'algorithme de maximisation de l'influence basé sur la réduction de l'espace de recherche dans les réseaux sociaux

3.3 Partie 3

3.3.1 Les réseaux sociaux

Dans cette section, on s'intéresse à l'analyse de réseaux à partir de la matrice de similarité finale dans les logiciels spécialisés que sont les logiciels de visualisation des réseaux sociaux *Gephi* et *SocNetV*. A cet effet, après l'avoir enregistré au format .csv en ajoutant les identifiants utilisateurs comme indices et noms de colonnes, la matrice de similarité finale a été importée dans ces logiciels pour construire des graphes puis réaliser une détection de communautés.

Logiciel GEPHI :

1. Graphe généré sur GEPHI

Le graphe a été généré dans *GEPHI*. Étant donné que la matrice de similarité est symétrique, nous avons choisi de créer un graphe de type *Non Dirigé*. De plus, pour éviter l'auto-similarité, nous avons remplacé les diagonales principales des matrices de similarité (*cosinus*, *Pearson* et *Jaccard*) par zéro, ce qui correspond à la valeur 1 dans la matrice finale. Dans le graphe, nous avons désactivé les *boucles* pour éviter les liens reliant un noeud à lui-même. Le graphe non orienté ainsi généré comporte 371 490 liens pour 610 noeuds. La spatialisation a été effectuée en utilisant l'**algorithme Force Atlas 2**. Pour représenter les noeuds, nous avons utilisé le degré pondéré (*Average Weighted Degree*) pour attribuer une taille proportionnelle au degré et une couleur variant du *Vert foncé* au *Cyan* jusqu'à l'*Orange* en fonction du degré pondéré. De plus, nous avons essayé d'appliquer un *filtre topologique* pour exclure les liaisons avec un poids trop faible (égal à 1). Mais cela n'ayant pas apporté de gain significatif, nous avons retiré ce filtre. Dans les graphes générés, les étiquettes sont positionnées à l'intérieur des noeuds, ce qui peut les rendre moins visibles sur l'image, mais cela contribue à l'esthétique des réseaux.

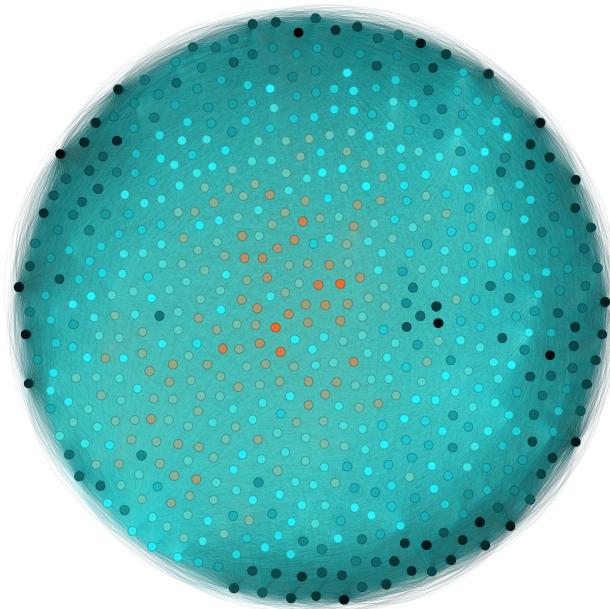


FIGURE 3.17 – Graphe non orienté généré sur *GEPHI* (avec coloration des noeuds selon la valeurs des degrés pondérés)

2. Détection des communautés sur GEPHI

Pour détecter les communautés dans *GEPHI*, nous avons utilisé l'*algorithme Modularité* [3]. Cet algorithme a permis d'identifier quatre (04) communautés, qui sont représentées par des couleurs distinctes qui sont dans l'ordre décroissant de la proportion de noeuds : 1-Rose (34,43%) ; 2-Vert (22,95%) ; 0-Orange (22,79%) et 3-Cyan (19,84%) comme l'illustre la figure 3.18 ci-contre.

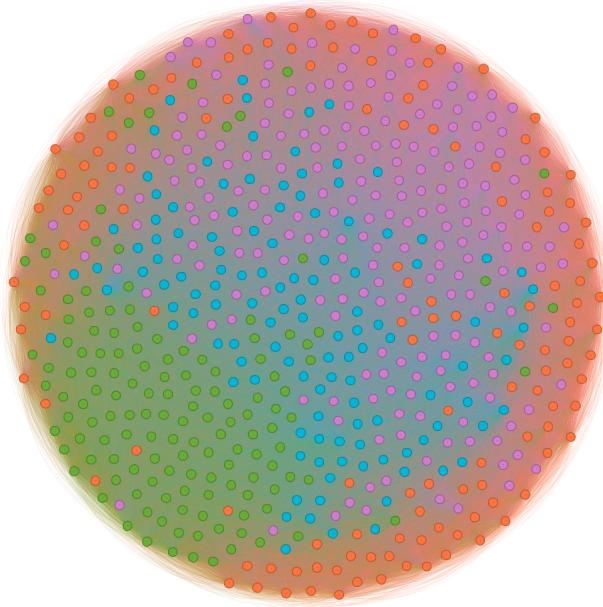


FIGURE 3.18 – Communauté détection dans le Graphe à l'aide l'algorithme modularité implémenté sur *GEPHI*

Chapitre 4

Conclusion et perspectives

L'objectif de ce projet est d'analyser les opinions des utilisateurs sur les films en utilisant un échantillon de données d'évaluations de films par un certain nombre d'utilisateurs. Pour ce faire, nous avons tout d'abord effectué des analyses de similarité à l'aide de matrices de similarité calculées entre les utilisateurs sur la base de leurs évaluations de films. Dans un second temps, sur la base des similarités calculées, nous avons étudié les relations entre les utilisateurs et mis en évidence les utilisateurs clés et les communautés, c'est-à-dire les groupes d'utilisateurs similaires, dans l'échantillon utilisé. Ces analyses ont permis d'identifier une matrice de similarité (celle de *Jaccard*) pour l'ensemble des données étudiées (*cosinus*, *Pearson* et *Jaccard*) en termes de performance dans un modèle de prédiction prédefini. Par ailleurs, une analyse des graphes non orientés générés à partir d'une matrice finale construite par agrégation de celles étudiées dans la première partie nous a permis d'identifier des utilisateurs clés, notamment ceux ayant un maximum de voisins ou ceux reliant un maximum de communautés, sur lesquels nous pouvons baser notre suivi de la diffusion de l'information sur le réseau social. Enfin, une analyse plus détaillée à l'aide de logiciels spécialisés dans la visualisation des réseaux sociaux nous a permis de détecter des communautés dans le réseau formé par le graphe de la matrice de similarité finale, notamment cinq (05) communautés identifiées à l'aide de l'algorithme de [modularitÃ©] [3] implémenté sur le logiciel *GEPHI*. Ces analyses nous ont permis de mieux comprendre les relations d'association et d'information entre les utilisateurs, tout en posant les bases de la mise en place d'un système de recommandation. En particulier, en exploitant les propositions faites pour par exemple identifier le plus petit ensemble de films pouvant atteindre tous les genres et styles que l'utilisateur a déjà regardés, ou pour recommander un film à un utilisateur en s'assurant que les films recommandés appartiennent à des types différents, etc. Ce projet démontre l'application pratique des techniques de similarité et de recommandation pour analyser les avis des utilisateurs sur les films. Il met en évidence la valeur de l'analyse des données pour fournir des recommandations personnalisées, ce qui peut être appliquÃ© dans divers domaines tels que l'industrie du divertissement, le commerce électronique et la personnalisation des services.

Bibliographie

- [1] G. ADOMAVICIUS et A. TUZHILIN. “Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. IEEE Transactions on Knowledge and Data Engineering”. In : (2005). URL : <https://ieeexplore.ieee.org/document/1423975> (visité le 29/05/2023).
- [2] M. BASTIAN, S. HEYMANN et M. JACOMY. “Gephi: an open source software for exploring and manipulating networks. International AAAI conference on weblogs and social media”. In : (2009). URL : <https://ojs.aaai.org/index.php/ICWSM/article/view/13937> (visité le 29/05/2023).
- [3] V. D. BLONDEL et al. “Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment”. In : (2008). URL : <https://iopscience.iop.org/article/10.1088/1742-5468/2008/10/P10008/meta> (visité le 29/05/2023).
- [4] M. CRUCIANU et al. “Cours - Fouille de graphes et réseaux sociaux (2)”. In : (2022). URL : <https://cedric.cnam.fr/vertigo/Cours/RCP216/coursFouilleGraphesReseauxSociaux2.html> (visité le 29/05/2023).
- [5] M. JACOMY et al. “ForceAtlas2, a Continuous Graph Layout Algorithm for Handy Network Visualization Designed for the Gephi Software. PLoS ONE”. In : (2014). URL : <https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0098679> (visité le 29/05/2023).
- [6] Dimitris KALAMARAS. “Social Network Visualizer 3.0.2 Manual”. In : (2021). URL : <https://socnetv.org/docs/index.html> (visité le 29/05/2023).
- [7] Y. KOREN, R. BELL et C. VOLINSKY. “Matrix factorization techniques for recommender systems. Computer”. In : (2009). URL : <https://ieeexplore.ieee.org/document/5197422> (visité le 29/05/2023).

- [8] J. LESKOVEC, K. J. LANG et M. MAHONEY. “Empirical comparison of algorithms for network community detection. In Proceedings of the 19th international conference on World wide web”. In : (2010). URL : <https://arxiv.org/abs/1004.3539> (visité le 29/05/2023).
- [9] Elsa NEGRE. “Les systèmes de recommandation : une catégorisation”. In : (2018). URL : <https://interstices.info/les-systemes-de-recommandation-categorisation/> (visité le 29/05/2023).
- [10] Romain PICOT-CLEMENTE. “Les systèmes de recommandations”. In : (2015). URL : <https://fr.slideshare.net/rpicotcl/presentation-decide-21-072015> (visité le 29/05/2023).
- [11] B. SARWAR et al. “Item-Based Collaborative Filtering Recommendation Algorithms. Proceedings of the 10th International Conference on World Wide Web”. In : (2001). URL : <https://dl.acm.org/doi/10.1145/371920.372071> (visité le 29/05/2023).
- [12] Christine THOËR, Stéfany BOISVERT et Katharina NIEMEYER. “La télévision à l’ère des plateformes”. In : (2022). URL : <https://doi.org/10.4000/questionsdecommunication.29183> (visité le 29/05/2023).