République Algérienne Démocratique et Populaire

Ministère de l'enseignement supérieur et de la recherche scientifique

Université de 8 Mai 1945 – Guelma -

Faculté des Mathématiques, d'Informatique et des Sciences de la matière

Département d'Informatique

Mémoire de Fin d'études Master

Filière : Informatique

Option : Systèmes Informatiques

Thème :

Système de recommandation basé sur la détection de communautés

Encadré Par :

Dr. Abdelmoumène Hiba

Présenté par :

Garnine Nardjes

Octobre 2020

Remerciement

A l'issue de ce travail, je remercie, en premier lieu, le bon Dieu de m'avoir donné la force et le courage de le mener à terme.

Je voudrai présenter mes remerciements à ma directrice de mémoire «Dr. Abdelmoumène Hiba».

Je voudrai également lui témoigner ma gratitude pour sa patience et son soutien qui m'ont été précieux afin de mener mon travail à bon port.

Et pour ses précieux conseils et son aide durant toute la période du travail.

Aussi je remercie messieurs les membres de jury et nos professeurs pour la qualité de l'enseignement qu'ils nous ont prodigué au cours de ces cinq années Passées au département d'informatique.

Enfin je tiens, également à exprimer ma sincère reconnaissance et ma profonde gratitude à

Ma famille, qui m'a toujours donné la possibilité de faire ce que je veux et qui ont toujours cru en moi.

Et à tous ceux qui ont contribué de près ou de loin à la réalisation de ce mémoire.

Dédicace

Tous les mots ne sauraient exprimer la gratitude, l'amour,

Le respect, la reconnaissance, c'est tout simplement

que je dédie ce mémoire.

À mes parents car ils m'ont inculqué un esprit de persévérance

et qu'ils m'ont toujours poussé et motivé dans mes études.

Sans eux, certainement je ne serais pas à ce niveau.

Ils ont tout fait pour mon bonheur et ma réussite.

Nulle dédicace ne puisse exprimer ce que je leur dois.

Que dieu leur réserve la bonne santé et une longue vie.

À mes frères et sœurs, qu'ils restent pour moi des exemples de persévérance, de courage et de générosité.

Et à tous ceux qui me sont chers et qui ont contribué à mes encouragements.

Résumé

Ce projet se situe dans le cadre du domaine des systèmes de recommandation (SR). Un système de recommandation est un outil de recherche d'information et de filtrage qui vise à proposer aux utilisateurs des items qui pourraient les intéresser. La plupart des solutions des SR se basent sur l'analyse des préférences des utilisateurs et leurs évaluations implicites ou explicites pour les items. Les différentes évaluations (ou appelées votes) sont souvent représentées sous forme d'une matrice utilisateurs x items. L'objectif de recommandation consiste à prévoir les évaluations manquantes dans cette matrice ou autrement dit recommander aux utilisateurs des items qui sont appréciés par leurs voisins.

Nous nous intéressons dans ce projet à appliquer les techniques d'analyse de réseaux complexes et plus précisément les techniques de détection de communautés dans le calcul de recommandation. En effet, la matrice des évaluations peut être vue comme la matrice d'adjacence d'un graphe biparti et ainsi les techniques de détection de communautés présenteront une bonne solution afin de détecter les utilisateurs et les items similaires.

L'objectif est de réaliser un système de recommandation en se basant sur les techniques de détection de communautés afin de palier à certains problèmes liés à la technique du filtrage collaboratif sur laquelle nous focalisons notre travail. Nous proposons une architecture qui combine le filtrage collaboratif et la détection de communauté.

Mots clés : Détection de communautés, Système de recommandation, Filtrage collaboratif, Graphe.

Table des matières

Liste des figures

Liste des tableaux

Introduction générale

Les Systèmes de Recommandation (SR) sont une plateforme pour interaction sociale, c'est une manière de proposer à l'utilisateur des produits qui sont susceptibles de l'intéresser. Ce dernier voit ainsi réduit son temps de recherche mais reçoit également des suggestions de la part du système auxquelles il n'aurait pas prêté attention.

En effet, les utilisateurs reçoivent des recommandations d'items (ressources) sur la base de leurs profils. Le profil d'un utilisateur évolue au cours du temps grâce aux interactions entre l'utilisateur et le système, notamment grâce aux évaluations produites par cet utilisateur. Ces évaluations sont sous la forme de notes explicites ou implicites.

Une des techniques les plus répandues des SR et qui reposent principalement sur les notes explicites données par les utilisateurs aux différentes ressources est le Filtrage Collaboratif (FC). Cette méthode base ses calculs sur l'aspect de collaboration et démarre avec ce qu'on appelle une matrice de note $Utilisateur \times Item$, calcule une similarité entre les utilisateurs/items pour désigner un ensemble de voisins et calculer, par la suite, une prédiction pour les notes manquantes. Cependant, les SR et spécialement la technique de filtrage collaboratif souffrent de plusieurs limites. Un problème important est lié à la matrice d'évaluation qui est la base du calcul. En effet, cette matrice est, en général, creuse et souffre de la rareté des données (parcimonie – *sparsity*). Cela rend le calcul de la similarité entre les utilisateurs imprécis et réduit par conséquent la précision des algorithmes de FC.

Pour surmonter les faiblesses des techniques basées sur la mémoire, un axe de recherche s'est concentré sur les techniques de clustering notamment la détection de communautés basées sur des modèles dans le but de rechercher des méthodes plus efficaces. Basées sur les notations, ces techniques regroupent les utilisateurs ou les items en clusters offrant ainsi une nouvelle façon d'identifier le voisinage.

Nous nous intéressons, dans notre travail, à l'utilisation des méthodes de détection de communautés afin de pouvoir trouver le voisinage d'un utilisateur/item. En effet, la détection de communautés peut être vue comme une généralisation du principe du filtrage collaboratif où le principe est de recommander à une personne les items les bien évalués par les membres de sa communauté. La

matrice utilisateur-item *R* peut être vue comme une matrice d'adjacence d'un graphe biparti. Ce graphe biparti comprend l'ensemble des utilisateurs et l'ensemble des items. Ils sont liés par des notes qui représentent le niveau d'appréciation d'un item par un utilisateur.

Notre objectif consiste alors, à trouver, dans une première étape, les communautés des utilisateurs (items) similaires en se basant sur les techniques de détection de communautés. La seconde étape consiste à prédire les notes manquantes afin de recommander des items aux utilisateurs, ceci se fait en se basant sur les communautés appropriées au lieu de calculer les prédictions sur tout le graphe.

Ce mémoire est organisé comme suit :

Chapitre 1 : nous présentons, dans ce chapitre, les fondements théoriques de notre travail, à savoir les systèmes de recommandation et la détection de communautés.

Chapitre 2 : ce chapitre est consacré à la conception de notre système de recommandation. Nous commençons par motiver et justifier nos choix, par la suite, nous détaillons chaque phase de réalisation de notre travail.

Chapitre 3 : nous présentons dans ce chapitre les détails d'implémentation de notre système. Ensuite, nous discutons les résultats trouvés et les évaluations élaborée

Chapitre 1 : Etat de l’art

1.1. Introduction

Dans ce chapitre, nous allons présenter les fondements théoriques de notre travail. Il est partagé en trois parties. Dans la première, nous allons présenter les systèmes de recommandation ainsi que leurs différentes techniques. Dans la deuxième partie, nous donnons un aperçu sur les structures communautaires et les techniques de détection de communautés. A la troisième partie, nous présentons une synthèse de quelques travaux de littérature qui se sont intéressés à l’application des méthodes de détection de communauté dans le cadre de la recommandation.

1.2. Les systèmes de recommandation

1.2.1. Définition et concepts de base

Les systèmes de recommandation sont une forme spécifique de filtrage d’information visant à présenter les éléments d'information qui sont susceptibles d'intéresser l'utilisateur. Ils peuvent être définis comme des programmes qui tentent de recommander les articles (produits ou services) les mieux adaptés à des utilisateurs particuliers (particuliers ou entreprises) en prévoyant l'intérêt d'un utilisateur pour un article en fonction d'informations connexes sur les articles, les utilisateurs et les interactions entre eux (Bobadilla et al., 2013).

L’élaboration de systèmes de recommandation a pour objectif de réduire la surcharge d’informations en récupérant les informations et les services les plus pertinents à partir d’une quantité énorme de données, fournissant ainsi des services personnalisés. La caractéristique la plus importante d'un système de recommandation est sa capacité à « deviner » les préférences et les intérêts d'un utilisateur en analysant le comportement de cet utilisateur et / ou celui d'autres utilisateurs pour générer des recommandations personnalisées (Resnick & Varian, 1997).

Les systèmes de recommandation ont été définis de plusieurs façons. La définition la plus populaire et la plus générale que nous citons ici est celle de Robin Burke (Burke, 2002) :

Un système de recommandation est : *« un système capable de fournir des recommandations personnalisées ou permettant de guider l'utilisateur vers des ressources intéressantes ou utiles au sein d'un espace de données important »*.

Dans tout système de recommandation, il existe deux entités qui représentent le cœur du système. Toute techniques de recommandation tourne autour de ces entités qui sont : les utilisateurs et les items.

L'utilisateur est la personne qui interagit avec le système, à qui le système recommande des items et à son tour donne son avis sur les items.

L'item est le terme général utilisé pour désigner la ressource que le système suggère aux utilisateurs.

Avec l'utilisateur et l'item, le domaine d'information d'un système de recommandation contient aussi une préférence exprimée par un utilisateur pour un item est appelée note, et est souvent représentée par un triplet (utilisateur ; item ; note). Ces notes peuvent prendre différentes formes. Cependant, la majorité des systèmes utilisent des notes sous formes d'une échelle de 1 à 5, ou bien des notes binaires (j'aime/je n'aime pas). L'ensemble des triplets (utilisateur ; item ; note) forme ce que l'on appelle la matrice des notes. Les paires (utilisateur ; item) où l'utilisateur n'a pas donné de note pour l'item sont des valeurs non connues dans la matrice.

Dans ce chapitre, nous utiliserons les notations suivantes concernant les différents éléments du modèle d'un système de recommandation :

On définit U l'ensemble des utilisateurs du système, et I l'ensemble des items. Les notes des utilisateurs sont stockées dans une matrice $R \in R_U \times R_I|$. I_u est l'ensemble des items notés par l'utilisateur u et U_i est l'ensemble des utilisateurs ayant notés l'item i. On désigne la note que l'utilisateur u a donné à l'item i par $r_{u,i}$. On note par $\bar{r}_u$ la moyenne des notes données par l'utilisateur u sur les items qu'il a noté (équation 1) et par $\bar{r}_i$ et par la moyenne des notes reçues par l'item i (équation 2).

$$\bar{r}_u = \frac{\sum_{i \in I_u} r_{u,i}}{|I_u|} \tag{1.1}$$

$$\bar{r}_i = \frac{\sum_{i \in U_i} r_{u,i}}{|U_i|} \tag{1.2}$$

1.2.2. Classification des systèmes de recommandation

Les techniques de recommandation peuvent être classées de différentes manières. Parfois plusieurs termes sont utilisés pour désigner une même méthode ou approche. La classification la plus utilisée reposent sur trois types : filtrage basé sur le contenu, filtrage collaboratif et filtrage hybride (Adomavicius & Tuzhilin, 2005). En plus de ces deux approches, Robin Burke (Burke, 2002) propose de considérer trois autres approches : la recommandation basée sur les données démographiques, la recommandation basée sur la connaissance (knowledge-based) et la recommandation basée sur l'utilité (utility-based). Mais il note que ces trois approches sont des cas particuliers des approches classiques.

L'objectif ici est de s'appuyer sur la classification la plus connue sur laquelle nous basons notre étude. Nous présentons dans la suite les approches basées contenu et le filtrage collaboratif, puis les approches hybrides (figure 1.1). Nous allons détailler la technique de filtrage collaboratif que nous allons utiliser dans notre travail.

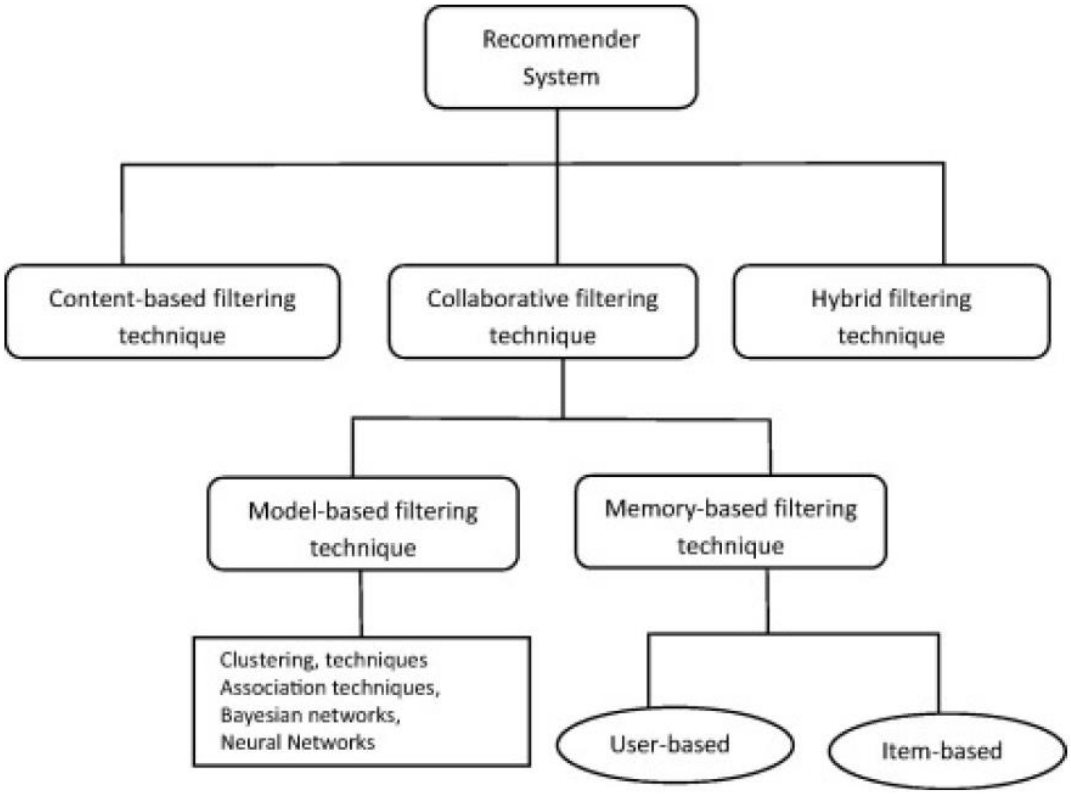


Figure 1.1 : *Classification des systèmes de recommandation (Isinkaye et al., 2015).*

1.2.2.1. Filtrage basé sur le contenu

Le système de recommandation basé contenu est un système de recherche en utilisant la description similaire entre les items qui vont être recommandés à l'utilisateur courant et les items qu'il a appréciés dans le passé , par exemple dans une recommandation des films le système basé contenu prend en considération les point communs comme : acteurs spécifiques, réalisateurs, genres de film, etc. (Adomavicius & Tuzhilin, 2005).

L'avantage des systèmes basés contenu est qu'ils permettent d'associer des documents à un profil utilisateur. Notamment, en utilisant des techniques d'indexation et d'intelligence artificielle. L'utilisateur est indépendant des autres ce qui lui permet d'avoir des recommandations même s'il est le seul utilisateur du système pour éviter certaines limitations et inconvénients des systèmes basés sur le contenu et collaboratifs (Belloui, 2008).

1.2.2.2. Filtrage collaboratif

Ce type de filtrage recueille des informations sur un utilisateur en lui demandant de noter les items et de formuler des recommandations basées sur les items les mieux notées par des utilisateurs qui ont un goût similaire. Les approches de FC font des recommandations en se basant sur les évaluations des items par un ensemble d'utilisateurs (voisins) dont les profils de notation sont les plus similaires à ceux de l'utilisateur cible. Ces méthodes supposent que si des utilisateurs ont les mêmes préférences sur un ensemble d'items, alors ils auront probablement les mêmes préférences sur un autre ensemble d'items qu'ils n'ont pas encore notés. Les méthodes du filtrage collaboratif peuvent être regroupées en deux catégories générales : *à base de mémoire (heuristique, voisinage proche) et à base de modèles (figure 1.1).*

1.2.2.2.1. Filtrage à base de mémoire

Dans la *collaboration à base de mémoire,* les notes des utilisateurs pour les items qui sont stockés dans le système sont directement utilisées pour prédire les notes des items non notés. Ce type de filtrage passe par deux phases importantes : le calcul de la similarité et le calcul de la prédiction de la note.

- La similarité : le calcul de la similarité se fait soit sur l'ensemble des utilisateurs (Resnick & Varian, 1997) soit sur l'ensemble des items (Sarwar et al., 2001) en utilisant des mesures de similarité et de distance, les plus utilisées dans la littérature sont *la similarité de cosinus,* ou *la corrélation de Pearson.*

- La prédiction : il s'agit de prédire la note des items non notés par les utilisateurs en se basant sur les notes données par ses voisins.

Nous donnons dans ce qui suit les détails de calcul de ces deux mesures (similarité cosinus que nous allons utiliser dans ce travail et la prédiction) en se basant sur les utilisateurs et en se basant sur les items.

1.2.2.2.1.1. Filtrage collaboratif basé utilisateurs

Le filtrage collaboratif basé sur les utilisateurs a été introduit pour la première fois dans le système GroupLens (Resnick & Varian, 1997), son principe de fonctionnement est très simple : déterminer les utilisateurs qui sont similaires à l'utilisateur courant, puis calculer une valeur de prédiction pour chaque item candidat à la recommandation en analysant les notes que les voisins de l'utilisateur courant ont exprimées sur cet item.

- *Calcul de la similarité cosinus*

La similarité entre deux utilisateurs u et u' est donnée par :

$$sim(u,u') = \cos(v_u, v_{u'}) = \frac{\sum_{i \in I_{uu'}} r_{u,i} \times r_{u',i}}{\sqrt{\sum_{i \in I_{uu'}} r_{u,i}^2} \cdot \sqrt{\sum_{i \in I_{uu'}} r_{u',i}^2}} \quad (1.3)$$

Où $v_u \; et \; v_{u'}$ représentent les vecteurs des utilisateurs $u \; et \; u'$respectivement et il s'agit ici de : $r_{u,i} \; et \; r_{u',i}$ respectivement.

Le cosinus varie entre 0 et 1. Une valeur égale à 1 indique que les deux utilisateurs ont des préférences identiques, une valeur égale à 0 indique qu'ils n'ont rien en commun. Un inconvénient majeur de l'utilisation du cosinus dans le filtrage collaboratif est qu'il ne tient pas compte de la variation dans le jugement des utilisateurs.

- *Calcul de la prédiction*

Le calcul de la prédiction, p est donné par la formule (1.4) :

$$p(u,i) = \frac{\sum_{k \in N(u) \cap U_i} sim(k,u) \times r_{k,i}}{\sum_{k \in N(u) \cap U_i} |sim(k,u)|} \quad (1.4)$$

Où, $N(u)$ est l'ensemble des voisins de l'utilisateur u, U_i est l'ensemble des utilisateurs ayant notés l'item i.

1.2.2.2.1.2. Filtrage collaboratif basé items

Le filtrage collaboratif à base d'items a été introduit par (Sarwar et al., 2001). La prédiction de la note de l'utilisateur u pour un item candidat i est calculée à partir de ses notes pour les items voisins (similaires) de i. Son principe de fonctionnement est le suivant : pour l'item i candidat à la recommandation, on détermine les voisins les plus proches (les items similaires) en calculant sa similarité avec les autres items disponibles et on calcule ensuite la prédiction de la note de l'utilisateur courant u pour l'item i à partir des notes que u a attribué aux voisins de i.

- *Calcul de la similarité cosinus*

La similarité entre deux items i et j est donnée par :

$$sim(i, j') = \cos(v_i, v_j) = \frac{\sum_{u \in U_{ij}} r_{u,i} \times r_{u,j}}{\sqrt{\sum_{u \in U_{ij}} r_{u,i}^2} \cdot \sqrt{\sum_{u \in U_{ij}} r_{u,j}^2}} \quad (1.5)$$

- *Calcul de la prédiction*

La prédiction de note qu'un utilisateur u peut donner à un item i est donnée par :

$$p(u, i) = \frac{\sum_{j \in I_u} sim(i, j) \times r_{u,j}}{\sum_{j \in I_u} |sim(i, j)|} \quad (1.6)$$

1.2.2.2.2. Filtrage collaboratif basé sur un modèle

Contrairement aux algorithmes basés sur la mémoire, seules les évaluations faites par l'utilisateur sont prises en compte dans les algorithmes basés sur le modèle afin de construire un modèle de prédiction des évaluations.

Ainsi, il faut d'abord créer des modèles qui ressemblent aux comportements des utilisateurs et ensuite, utiliser ces modèles afin d'émettre les recommandations. En pratique, il a été démontré que l'approche par mémoire offre de meilleures performances en termes de précision alors que l'approche par modèle est plus efficace à grande échelle avec de gros ensembles de données.

1.2.2.3. Filtrage hybride

Les systèmes de filtrage hybrides combinent le système de filtrage collaboratif avec d'autres techniques de recommandation (généralement avec des systèmes basés sur le contenu) pour faire

des prédictions ou des recommandations pour éviter certaines limitations et inconvénients des systèmes basés sur le contenu et collaboratifs (Rao & Talwar, 2008).

En général, le filtrage hybride utilise certaines méthodes pour combiner les ensembles de recommandations telles que la pondération, la cascade, la commutation, etc. afin de produire des recommandations finales pour les utilisateurs (Burke, 2002).

1.2.3. Evaluation des systèmes de recommandation

Evaluer un système de recommandation permet de mesurer ses performances vis-à-vis de ses objectifs. De ce fait, le choix des mesures à utiliser diffère selon les objectifs fixés. La base initiale est partagée en deux groupes de données. Le premier va servir de base d'apprentissage et le deuxième va servir de base de test. On prend chaque individu u de la base de test, on essaie de prédire sa note pour l'item i à partir des notes des individus de la base d'apprentissage et de ses notes disponibles pour ses autres items (Ziani, 2018).

- La mesure de l'erreur absolue moyenne (EAM ou MAE pour Mean Absolute Error en anglais) qui est la mesure la plus employée dans les systèmes de recommandation, elle estime la moyenne de la différence absolue entre les évaluations et les prédictions. Le système de recommandation collaboratif est jugé performant quand la valeur de MAE est petite. Cette mesure est donnée par l'équation suivante :

$$MAE = \frac{\sum_{(u,i)\in K}\left|r_{u,i} - \widehat{r_{u,i}}\right|}{|K|} \qquad (1.7)$$

$r_{u,i}$ est la vraie note donnée par u à i.

$\widehat{r_{u,i}}$ est la note prédite par le SR.

K est l'ensemble des couples (utilisateur, item) pour lesquels la confrontation est effectuée.

- RMSE (Root Mean squared error)

$$RMSE = \frac{\sqrt{\sum_{(u,i)\in K}(r_{u,i} - \widehat{r_{u,i}})^2}}{|K|} \qquad (1.8)$$

1.2.4. Limites des systèmes de recommandation

Avec l'avènement du web et la masse importante d'information que contient, les systèmes de recommandation présentent un outil très important avec beaucoup d'avantages afin de faire face

au problème de surcharge d'information et ainsi trouver l'information nécessaire pour les utilisateurs. Cependant, ces systèmes présentent un certain nombre de limites, on va citer dans ce qui suit deux importants problèmes de ces systèmes.

- *Démarrage à froid*, c'est un problème qui se pose avec l'arrivée d'un nouvel élément dans le système de recommandation soit un utilisateur soit un item. Le système doit le gérer comme un cas spécial, le problème de démarrage à froid c'est qu'on n'a pas assez d'informations pour émettre des recommandations (Rashid et al., 2002).

- *Parcimonie (Sparsity – rareté des données),* le problème de rareté est lié à l'indisponibilité d'un grand nombre d'items notés pour chaque utilisateur actif. Le nombre d'items candidats à la recommandation est souvent énorme et les utilisateurs ne notent qu'un petit sous-ensemble des items disponibles. De ce fait, la matrice des notes est une matrice creuse avec un taux de valeurs manquantes pouvant atteindre 95% du total des valeurs (Papagelis et al., 2005). Par exemple, sur Amazon, si les utilisateurs actifs peuvent avoir acheté 1% des articles et que le montant total des articles est d'environ 2 millions de livres, cela signifie qu'il n'y a que 20 000 livres évalués. Par conséquent, une telle rareté dans les évaluations dégrade la sélection précise des voisins dans l'étape de formation du quartier et conduit à de mauvais résultats de recommandation (Adomavicius & Tuzhilin, 2005; Linden et al., 2003).

1.3. La détection de communautés

Nous commençons par présenter quelques notions liées à la théorie des graphes. Par la suite, nous présentons les techniques de détection de communautés.

1.3.1. Notions relatives à la théorie des graphes

Cette section rappelle quelques définitions classiques de la théorie des graphes (Gondron & Minoux, 1995; Newman, 2003), et introduit les notations utilisées dans ce mémoire.

1.3.1.1. Définitions

Un graphe G est défini comme un couple $G = (V, E)$ composé d'un ensemble de sommets V (vertices en anglais) et d'un ensemble d'arêtes E (edges) reliant certaines paires de sommets de V. Formellement, $E \subseteq V \times V$. Nous considérons le cas de graphes finis, c'est-à-dire ayant un nombre fini de sommets et d'arêtes et notons $n = |V|et\ m = |E|$ (Canu, 2017).

Dans ce qui suit, on donne les définitions de quelques concepts liés aux graphes.

- Incidence, est une arête $(v_i, v_j) \in E$ est dite incidente à v_i et à v_j et ces nœuds sont dits voisins ou adjacents.
- Degré, le degré d'un nœud v est le nombre d'arêtes incidentes à ce nœud, noté $d(v)$.
- Densité d'un graphe : Rapport entre le nombre d'arêtes observées et le nombre maximal d'arêtes possibles. $Densité = 0$, tous les sommets sont isolés ; $Densité = 1$, graphe complet, c.-à-d. il y a un lien entre chaque paire de sommets [1].

$$densité = \frac{2|E|}{|V| \cdot (|V| - 1)}$$

- Orienté, un graphe orienté $G = (V, E)$ est un graphe pour lequel : $\forall\ v_i, v_j \in V$, (v_i, v_j) est distinguable de (v_j, v_i). Autrement dit, il y a une signification des arêtes (extrémité initiale, extrémité finale). Dans un graphe orienté, on parle d'arc et non pas d'arête.
- Pondéré, un graphe pondéré $G = (V, w)$ est un ensemble de nœuds V et une fonction de poids $w: V \times V \to \mathbb{R}$ qui associe une valeur réelle à chaque couple de nœuds. Autrement dit, une valeur appelée poids est attribuée à chaque arête.

La figure 1.2 montre un exemple de graphe, le degré de ses nœuds et sa densité.

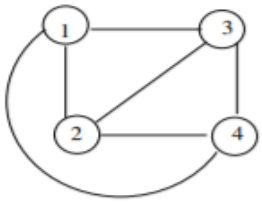


Figure 1.2 : *Exemple de graphe.*

Ce graphe possède 6 arêtes et chaque sommet du graphe est de degré 3. La densité = $\frac{2*6}{4*(4-1)}$ = 1

- Un graphe est dit simple, s'il ne comporte pas de boucle et comporte au plus une arête entre deux sommets
- Un graphe partiel est le graphe obtenu en supprimant certains arcs ou arêtes
- Un sous-graphe est le graphe obtenu en supprimant certains sommets et tous les arcs/arêtes incidents aux sommets supprimés.
- Un graphe est dit Complet si chacun de ses sommets est relié avec tous les autres sommets (Müler, 2012).

➢ Le chemin, suite des arrêtes dont l'extrémité final d'un sommet est l'extrémité initiale d'un autre sommet (Sigward, 2002).

➢ La distance, il probable qu'il existe plusieurs chemins entre une paire de sommets et que ces chemins diffèrent en longueur. Le chemin le plus court entre deux nœuds i et j est dit distance géodésique ou simplement distance. On note $d(i,j)$, elle compte le nombre d'arêtes entre le sommet i et le sommet j.

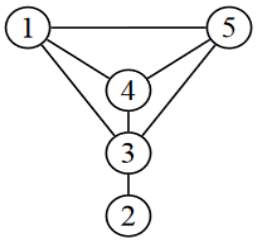
➢ Le diamètre, le diamètre d'un graphe quantifie la distance entre les deux nœuds les plus éloignés du graphe. C'est la plus grande distance géodésique entre n'importe quelle paire de nœuds, et elle est égale au maximum de $d(i,j)$.

Il est possible de représenter un graphe de façon matricielle à l'aide de trois types de matrices, matrice d'adjacence, matrice de degrés et matrice Laplacienne. Dans ce qui suit, nous allons donner les définitions des deux premiers types de matrices : adjacence et degré.

1.3.1.1.1. Matrice d'adjacence

La matrice d'adjacence A est une matrice carrée $n \times n$. Pour un graphe non orienté, la matrice d'adjacence est symétrique où $A_{i,j} = 1$ si i et j sont voisins, 0 sinon.

$$A_{i,j} = \begin{cases} 1, si\ i\ et\ j\ sont\ voisins \\ 0, sinon \end{cases}$$



$$M = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{pmatrix}$$

Figure 1.3 : *Exemple d'un graphe avec sa matrice d'adjacence.*

1.3.1.1.2. Matrice de degrés

La matrice des degrés est une matrice diagonale qui contient des informations sur le degré de chaque nœud, c'est-à-dire le nombre d'arêtes attachées à chaque nœud (Kumar et al., 2006; Scott, 2000).

$$D_{i,j} = \begin{cases} d(v_i) si\ i = j, où\ d(v_i) = \sum_{i=1}^{n} A_{i,j} \\ 0\ sinon \end{cases}$$

La matrice des degrés de l'exemple de la figure 1.3 est :

$$\begin{pmatrix} 3 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 3 \end{pmatrix}$$

1.3.1.2. Mesures de centralité

Les mesures de centralité visent à évaluer des propriétés souvent abstraites des entités du graphe. On distingue (Combe, 2014) :

- *Centralité de degré*

La mesure de la centralité la plus simple est le degré. Que le graphe soit pondéré ou pas, on mesure la somme de l'intensité de la connexion d'un sommet avec ses voisins directs. En effet, les nœuds centraux d'un graphe doivent être les plus actifs, avec un plus grand nombre de voisins.

$$C_D(n_i) = d(n_i) = \sum_j A_{i,j}$$

Cette mesure est indiquée dans les situations où on peut assimiler l'importance d'un sommet à son activité potentielle de communication.

- *Centralité d'intermédiarité*

La centralité d'intermédiarité (betweeness centrality) est une autre mesure de centralité d'un sommet dans un graphe. L'intermédiarité d'un sommet $u \in V$ est définie par :

$$C_B = \sum_{v,v' \in V, v \neq v'} \frac{\varphi(v, v'|u)}{\varphi(v, v')}$$

Où, $\varphi(v, v')$ est le nombre de plus courts chemins passant du sommet v au sommet v' et $\varphi(v, v'|u)$ est le nombre de plus courts chemins du sommet v au sommet v' passant par u.

Les sommets qui se trouvent fréquemment sur les plus courts chemins entre deux autres sommets ont une intermédiarité plus grande que les autres.

- *Edge betweeness*

L'intermédiarité peut également être définie pour une arête e.

$$C_{EB} = \sum_{(v,v') \in V \times V} \frac{\varphi(v, v'|e)}{\varphi(v, v')}$$

Où, $\varphi(v, v'|e)$ est le nombre de plus courts chemins du sommet v au sommet v' passant par l'arête e.

Dans ce qui suit, on donne la définition d'un type spécifique de graphe, le graphe biparti. Ce dernier sera utilisé dans la suite du mémoire.

1.3.1.3. Graphe biparti

Un graphe biparti (ou bigraphe) est un réseau dont les nœuds sont divisés en deux ensembles disjoints U et V ; les seuls liens dans le graphique sont ceux qui connectent un nœud U à un nœud V. Il n'y a pas de lien reliant deux nœuds U ou deux nœuds V.
La figure 1. Montre un graphe qui contient deux types de nœuds : les nœuds représentés en carrés verts et les nœuds représentés en des cercles bleues.

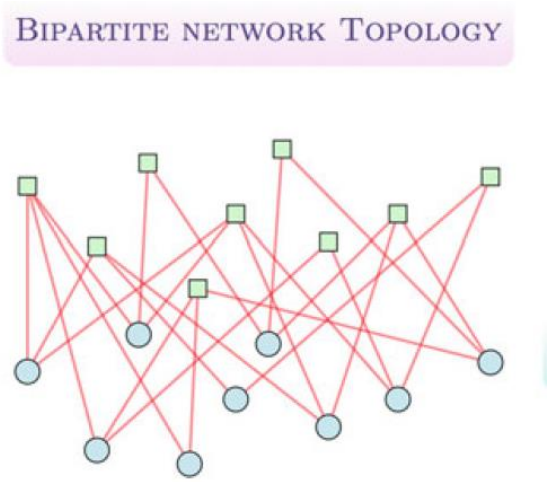


Figure 1.4 : *Exemple de graphe biparti (Alzahrani & Horadam, 2016)*

1.3.2. Structures communautaires

Dans cette section, nous allons étudier les structures communautaires dans les graphes ainsi que les différentes méthodes de leur détection et l'objectif derrière cette détection. Nous allons commencer par donner des définitions de ce qui est entendu par communauté dans la littérature. Par la suite, nous donnons un aperçu des différents algorithmes qui permettent de détecter de telles structures communautaires.

1.3.2.1. Définition d'une communauté

Le premier problème dans le regroupement de graphes est de rechercher une définition quantitative de la communauté. Aucune définition n'est universellement acceptée. En fait, la définition dépend souvent du système spécifique et / ou de l'application que l'on a à l'esprit.

(Radicchi et al., 2004) ont introduit deux définitions de communautés fondées sur la cohésion interne et externe. Pour qu'une communauté soit considérée comme forte, chaque nœud de la communauté doit être connecté à plus de nœuds à l'intérieur de la communauté qu'à l'extérieur, c'est à dire le degré interne de chaque nœud doit être supérieur à son degré externe. Une communauté répond à la définition de Radicchi lorsque la somme de tous les degrés internes est supérieure à la somme de tous les degrés externes.
Pour une revue exhaustive des définitions de communautés, voir (Santo Fortunato, 2010, 2016).

1.3.2.2. Approches de détection de communautés

C'est un processus de découverte des groupes cohésifs dans un graphe (réseau) (Bedi & Sharma, 2016), où on crée des partitions des nœuds (figure 1.) à partir des relations entre eux, c'est-à-dire les communautés possèdent des nœuds fortement connectés par rapport aux autres dans le graphe (Brandes et al., 2009; Lancichinetti & Fortunato, 2009; Newman, 2004a; Schaeffer, 2007).
Les applications de la détection de communautés sont nombreuses. C'est une étape souvent nécessaire pour nombre d'opérations de traitement de grands graphes notamment pour la visualisation, la compression et la parallélisation des calculs. Un autre champ d'applications important est le calcul de recommandations (Kanawati, 2013).

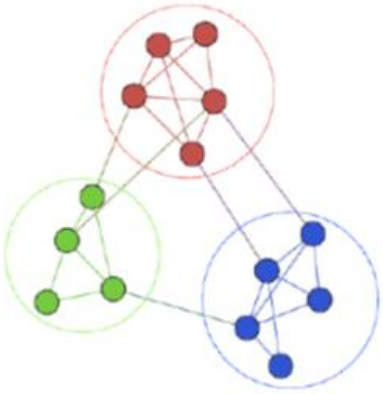


Figure 1.5 : *Exemple d'une structure communautaire dans un graphe.*

1.3.2.3. Fonctions de qualité

Afin de mesurer la qualité des communautés détectées, plusieurs fonctions de qualité ont été utilisées dans la littérature. Ces fonctions de qualités sont tirées, en général, du clustering et de la classification automatique. Dans cette section, nous allons citer quelques fonctions de qualité qui sont le plus utilisées dans le cadre de la détection de communautés, tout en consacrant une sous-

section pour une mesure de qualité qui a été introduite spécialement pour mesurer la qualité des communautés trouvées, à savoir la modularité.

(Yang & Leskovec, 2012) ont défini quatre caractéristiques qu'ils estiment désirables dans les communautés recherchées. La densité interne où les nœuds à l'intérieur de la communauté sont très connectés entre eux ; la séparabilité où la communauté présente des caractéristiques différentes de son entourage ; la cohésion interne, les caractéristiques de la communauté sont robustes à la suppression de nœuds ou d'arêtes ; et la fermeture triadique.

D'autres mesures existent telles que la Surprise (Aldecoa & Marín, 2013) et la Signifiance (Traag et al., 2013) sont basées sur le calcul d'une dissimilarité de distribution (Creusefond, 2016).

1.3.2.3.1. La modularité

La modularité a été proposée par Girvan et Newman (Girvan & Newman, 2002). Elle mesure la différence entre le nombre d'arêtes internes à la communauté (le premier terme) et l'espérance de cette valeur dans le modèle de configuration (le second terme). Le modèle de configuration correspond au même graphe dont les arêtes sont coupées en deux et ces demi-arêtes sont rattachées de manière aléatoire.

Ainsi, si un groupe d'individus à une densité de connexion significativement supérieure à ce que le modèle prévoit, ces connexions ne sont pas explicables par des branchements aléatoires.

La modularité quantifie la densité interne ainsi que la séparabilité. La densité interne est représentée par le nombre d'arêtes internes. On remarque aussi qu'une augmentation du nombre d'arêtes externes entraîne une diminution de la modularité, du fait de la comparaison avec le modèle. La modularité mesure donc aussi la séparabilité (Creusefond, 2016).

Elle est calculée par la formule suivante [1] :

$$Q(C) = \sum_{i,j \in C} A_{ij} - \frac{d_i d_j}{2m}$$

- m **:** nombre des arrêtes dans le graphe.
- $d_i d_j$: degré de centralité des nœuds v_i et v_j.
- C : la communauté étudiée.

Pour un graphe avec k communautés et m arêtes, la modularité est donnée par (Meghanathan):

$$Q = \sum_{l=1}^{k} \sum_{i \in C, j \in C} A_{i,j} - \frac{d_i d_j}{2m}$$

Plus la modularité est grande, meilleure est la partition.

1.3.2.4. Algorithmes de détection de communautés

Dans cette section, nous allons présenter une brève revue de littérature sur les approches de détection de communautés.

En fonction du principe méthodologique sous-jacent ainsi que la définition adoptée de la communauté plusieurs approches de détection de communautés ont été proposées. L'état de l'art de (Santo Fortunato, 2010) compte plus de 50 méthodes sans être exhaustif. L'article de (Xie et al., 2013) compare 10 méthodes. D'autres surveys plus récents comptent plusieurs algorithmes aussi entre anciens et récents (Bedi & Sharma, 2016; Cai & Lijia, 2016; El-moussaoui et al., 2019; Santo Fortunato, 2016; Khan & Niazi, 2016). Ces auteurs adoptent diverses classifications et catégorisations de ces algorithmes.

Comme il existe de nombreuses approches, nous allons retenir celles ayant reçu le plus d'intérêt de la part de la communauté scientifique. Ces approches illustrent aussi la diversité de méthodologies et donnent une vue d'ensemble des techniques proposées selon leurs principes méthodologiques. Il est également important d'adopter une catégorisation. La classification que nous allons adopter dans ce mémoire est la suivante :

- Algorithmes de classification hiérarchique,
- Algorithmes d'optimisation d'une fonction objective,
- Algorithmes à base de modèle.

1.3.2.4.1. Algorithmes de classification hiérarchique

Généralement, les algorithmes hiérarchiques (Parthasarathy et al., 2011) de clustering est un moyen d'extraction de communautés, qui a plus de succès dans la représentation des liens inter/intracommunautaire (Nettleton, 2013). Les méthodes hiérarchiques peuvent être divisées en deux méthodes principales en se basant sur la stratégie selon laquelle la hiérarchie ou la granularité

des communautés se construit depuis et vers la composition atomique (nœuds) du graphe (figure 1.6) : les méthodes hiérarchiques ascendantes (agglomératives) et les méthodes hiérarchiques descendantes (séparatives). Les deux méthodes sont basées sur la définition d'une mesure de similarité entre chaque paire de nœuds.

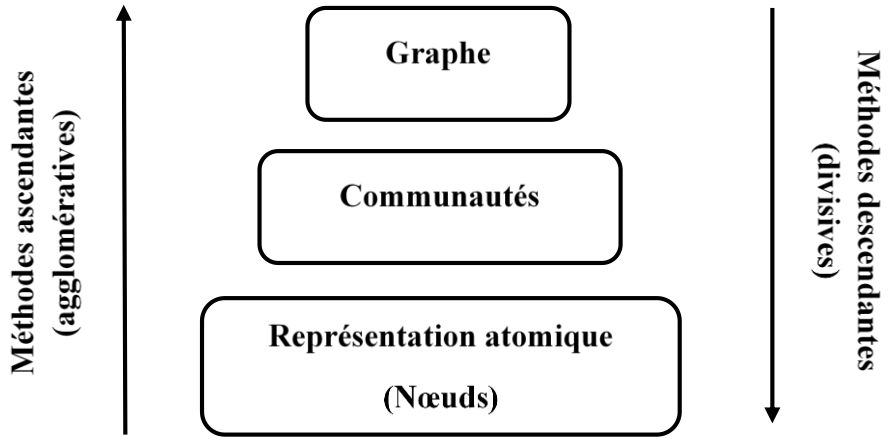


Figure 1.6 : *les méthodes hiérarchiques* (Hamadache, 2017).

- *Algorithmes agglomératifs*

Les algorithmes agglomératifs commencent avec chaque nœud du graphe dans sa propre communauté, et à chaque étape fusionnent les communautés jugées suffisamment similaires, en continuant jusqu'à ce que le nombre souhaité de communautés soit obtenu ou que les communautés restantes soient jugées trop différentes pour être fusionnées.

(Newman, 2004b) a adopté une méthode agglomérative pour son algorithme qui est un algorithme basé sur une optimisation gloutonne de la modularité. Cette métrique permet de chercher directement le découpage en communautés correspondant à la valeur maximale de la modularité pour un graphe donné. Initialement, chaque nœud est une communauté. Pour toutes les paires de communautés voisines, la modification de la modularité en cas de fusion est calculée et les deux communautés qui apportent le gain le plus important sont réunies dans une seule. La métrique utilisée dans cet algorithme est la *modularité* qui est calculée à chaque jointure de deux communautés.

Un des algorithmes agglomératifs les plus répandu aussi est l'algorithme de (Blondel et al., 2008) qui est basé aussi sur l'optimisation de la modularité. A chaque itération, l'algorithme tente de

maximiser la modularité. Comme dans la version de Newman, au début chaque nœud est mis dans une communauté différente. Les nœuds sont parcourus dans un ordre aléatoire et, pour chacun d'entre eux, on regarde si le placement dans la communauté d'un de ses voisins apporte un gain en modularité. Si c'est le cas, le nœud est déplacé dans cette communauté, sinon il reste dans l'ancienne communauté. Lorsqu'aucun gain n'est plus possible, la deuxième étape de l'algorithme commence. Un nouveau graphe est créé, dont les nœuds correspondent aux communautés de l'étape précédente, et la première étape est répétée sur ce graphe. Les itérations s'arrêtent au moment où une nouvelle étape n'apporte plus une croissance de la modularité, la figure 1.7 Illustre l'algorithme de Louvain (Nedioui, 2015).

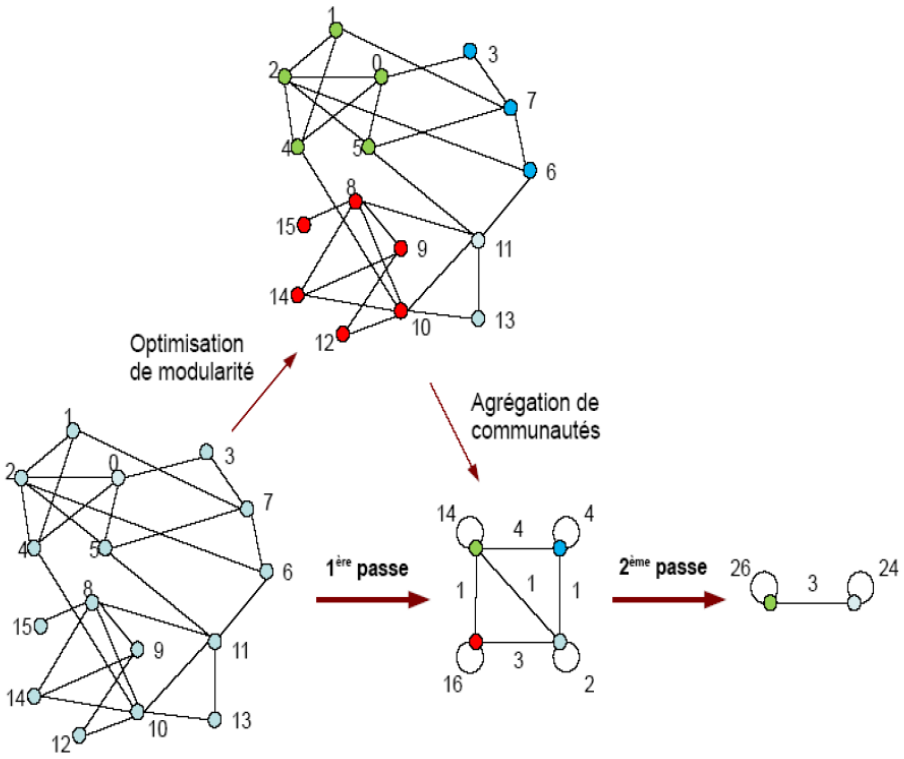


Figure 1.7 : *Illustration de l'algorithme de Louvain.*

- *Algorithmes divisifs*

Ces méthodes reposent sur l'identification des éléments du graphe (arêtes et sommets) qui sont positionnés entre les communautés (Papadopoulos et al., 2012). Par exemple, l'algorithme fondateur de Girvan et Newman (Girvan & Newman, 2002) supprime progressivement les arêtes d'un graphe en se basant sur l'*edge betweeness* jusqu'à ce que les communautés émergent comme des composants déconnectés. Plusieurs mesures d'*edge betweeness* entre les nœuds ont été

conçues, par exemple, la marche aléatoire et la centralité de l'information (Fortunato et al., 2004). Un principe similaire est adopté par les méthodes de suppression de nœuds (Vragovic & Louis, 2006) ; ces méthodes suppriment les sommets afin de révéler la structure de la communauté.

1.3.2.4.2. Algorithmes d'optimisation d'une fonction objective

Il existe un certain nombre d'algorithmes se basant sur des heuristiques pour définir la structure de communauté des réseaux. Ce type d'algorithme consiste à définir une fonction objective dont la valeur varie selon les communautés dégagées. La fonction est maximale pour la meilleure structure de communauté. A titre d'exemple on peut citer les travaux de (Agarwal & Kempe, 2008) et (Berry et al., 2009) qui tentent de maximiser la modularité. Une autre approche basée optimisation de la modularité est celle de CNM (Clauset-Newman-Moore) (Clauset et al., 2004). C'est un algorithme qui suit une stratégie de partitionnement hiérarchique où, à chaque étape il fournit la plus forte augmentation de la modularité. L'algorithme s'arrête après avoir trouvé une communauté unique.

1.3.2.4.3. Algorithmes basés modèle

Il s'agit d'une large catégorie de méthodes qui considèrent soit un processus dynamique se déroulant sur le graphe, qui révèle ses communautés, soit elles considèrent un modèle sous-jacent de nature statistique qui peut générer la division du réseau en communautés.

Des exemples de processus dynamiques sont la propagation d'étiquettes (Gregory, 2009; Leung et al., 2009; Raghavan & Kumara, 2007), la synchronisation des oscillateurs Kuramoto (Arenas et al., 2008), le flux de diffusion, mieux connu sous le nom d'algorithme de cluster de Markov (Van Dongen, 2000), et le modèle de spin populaire de (Reichardt & Bornholdt, 2006). De plus, la détection de communauté peut être présentée comme un problème d'inférence statistique (Hastings, 2006), en supposant un modèle probabiliste sous-jacent, tel que le modèle de partition plantée, qui génère la structure de la communauté et estime les paramètres de ce modèle.

D'autres approches basées sur des modèles reposent sur le principe qu'un bon clustering est déterminé par un faible coût d'encodage, ainsi elles effectuent une détection communautaire en trouvant la structure de cluster qui entraîne le coût d'encodage de cluster le plus bas possible (Chakrabarti, 2004; Rosvall & Bergstrom, 2008).

Nous allons donner dans le tableau 1.1 une classification et une comparaison de différents travaux en se basant sur certaines caractéristiques qui diffèrent chacun d'entre eux. Les travaux choisis ne représentent pas une liste exhaustive des recherches réalisées dans le domaine de la détection de communautés.

Les critères de comparaison sont :

- Approche : la méthodologie utilisée pour trouver les communautés,
- Fondement théorique et paramètres : le principe théorique et technique de l'approche appliquée,
- Type du graphe : pondéré ou non / orienté ou non,
- Nature du graphe : statique ou dynamique,
- Complexité : complexité de l'algorithme.

Référence	**Approche**	**Fondement théorique & paramètres**	**Type du graphe**	**Nature du graphe**	**Complexité**
(Xiang et al., 2017)	Divisive	Optimisation locale de la modularité	Non pondéré – Non orienté	Statique	
(Girvan & Newman, 2002)	Divisive	Edge betweenss	Pondéré – Non orienté	Statique	$O(m^2n)$
(Blondel et al., 2008)	Agglomérative	Optimisation de la modularité	Non pondéré – Non orienté		$O(m)$
(Raghavan & Kumara, 2007)	Label Propagation	Propagation des étiquettes itérative (voisins)	Non pondéré – Non orienté	Statique	$O(m)$
(Rosvall & Bergstrom, 2008)	Infomap	Random Walk	Pondéré – Orienté	Statique/ Dynamique	$O(m)$
(Fortunato et al., 2004)	Hiérarchique	Centralité	Non pondéré – Non orienté	Statique	$O(m^3n)$
(Clauset et al., 2004)	Hiérarchique – agglomérative	Optimisation de la modularité	Non pondéré – Non orienté	Statique	$O(mdlogn)$ d : la profondeur du dendrogramme
(Newman, 2016)	Divisive	Modularité	Non pondéré – Non orienté	Statique	$O((m+n)n)$

Tableau 1.1 : *synthèse de quelques techniques de détection de communautés.*

1.4. Détection de communautés et système de recommandation

Nous donnons dans cette section une classification de quelques travaux connexes en se basant sur un ensemble de critères :

Référence	Motivation	Type de recommendation	Données	Type du graphe	Nature du graphe	Approche de DC
(Mkhitaryan, 2019)	Parcimonie – Données manquantes	Collaboration	BDD créée artificiellement	Mono-parti	Statique	5 algorithmes de littérature
(Gorripati & Vatsavayi, 2017b)	Démarrage à froid – Parcimonie	FC	MovieLens	Mono-parti	Statique	Label Propagation
(Gasparetti et al., 2018)	Améliorer la précision – Démarrage à froid – Parcimonie	FC	Réseau social	Biparti	Statique	Approches existantes
(Xin et al., 2014)	Améliorer la précision	FC	BDD réelle	Mono-parti	Statique	Nouvel algorithme – Important Greedy
(Sahebi & Cohen, 2011)	Démarrage à froid	FC	BDD *imhonet*	Mono-parti	Statique	Principle Modularity Maximization
(Crawford & Cho, 2015)	Améliorer la performance et la précision	FC	BDD Beer-Advocate	Biparti	Statique	Belief Propagation, Louvain et CNM
(Nagwekar & Shirsat, 2017)	Démarrage à froid	FC	Réseau social	Mono-parti	Statique	Bron-Kerbosch algorithm
(Kamahara et al., 2005)	Trouver d'autres aspects des utilisateurs	FC	Développement d'un système	Multiplexe	Statique	Nouvel algorithme
(Falih et al., 2018)	Parcimonie	FC	MovieLens	Multiplexe	Statique	MuxLicod
(Gorripati & Vatsavayi, 2017a)	Améliorer la précision	Filtrage basé contenu	MovieLens	Mono-parti	Statique	Label Propgation

Tableau 1.2 : *Synthèse des travaux sur la recommandation et la détection de communautés.*

- Motivation : La raison derrière laquelle les auteurs ont opté pour les techniques de détection de communautés.
- Type de recommandation : filtrage collaboratif, filtrage basé contenu ou filtrage hybride.
- Données : le type de données utilisé pour valider le travail (une BDD, un réseau social, …).

- Type du graphe : si les auteurs ont traité un graphe mono-parti ou biparti ou multiplexe.
- Nature du graphe : statique ou dynamique.
- Approche de détection de communauté : l'algorithme ou l'ensemble des algorithmes utilisés pour détecter les communautés dans le graphe.

1.5. Conclusion

Dans ce chapitre, nous avons étudié les systèmes de recommandation ainsi que leurs types, et leurs problèmes, nous avons vu aussi les notions de base de la théorie des graphes, puis nous avons expliqué la détection de communauté et ses algorithmes, et on a terminé ce chapitre par quelques travaux sur la détection des communautés dans les systèmes de recommandation.

Nous allons présenter dans le chapitre suivant la conception de notre système de recommandation.

Chapitre 2 : Un système de recommandation basé sur la détection de communautés

2.1. Introduction

Dans ce chapitre nous allons décrire les étapes de réalisation de notre système de recommandation. Nous commençons d'abord par une présentation de notre problématique et l'objectif de notre approche. Par la suite, nous allons présenter l'architecture générale de notre système ainsi qu'une description détaillée de chaque phase de travail. Le chapitre présente également l'algorithme de détection de communauté adapté à notre cas.

2.2. Problématique et objectif de notre approche

Les systèmes de recommandation sont une forme spécifique de filtrage d'information visant à présenter les éléments d'information qui sont susceptibles d'intéresser l'utilisateur. Leurs solutions se basent sur l'analyse des préférences des utilisateurs et leurs évaluations (notes) pour les items. Les techniques utilisées sont classées en trois catégories : filtrage basé sur le contenu, filtrage collaboratif et filtrage hybride.

Dans ce travail, nous nous concentrons sur le filtrage collaboratif qui est considéré comme l'une des techniques les plus utilisées dans les systèmes de recommandation, en raison de son efficacité et de sa grande précision (Bobadilla et al., 2013). Cette technique repose sur une hypothèse fondamentale stipulant que les utilisateurs ayant noté, aimé les mêmes items ou ayant des comportements similaires (par exemple, regarder, acheter, …), dans le passé, ils évalueront ou agiront sur d'autres items de la même manière, dans le futur. En effet, Le filtrage collaboratif utilise les préférences connues d'un groupe d'utilisateurs pour faire des recommandations ou des prédictions sur les préférences inconnues pour d'autres utilisateurs.

Comme nous l'avons exposé dans le chapitre précédent, le processus de recommandation pour cette technique démarre lorsque les utilisateurs expriment leurs préférences en notant des items. Le

système analyse ces évaluations pour déterminer les préférences exactes de l'utilisateur, puis fait correspondre les préférences de l'utilisateur actif et la collection de préférences pour découvrir la catégorie d'utilisateurs ayant des goûts similaires à l'utilisateur actif. Enfin, le système recommande un ensemble d'items pour l'utilisateur actif en fonction des préférences de ses utilisateurs similaires. Ces données sont sous la forme d'une matrice utilisateur-item où chaque utilisateur a donné son degré d'intérêt (à travers une note) à un certain nombre items.

Dans un scénario typique de filtrage collaboratif, il existe une liste de n utilisateurs $\{u_1, u_2, \dots, u_n\}$ et une liste de m items $\{i_1, i_2, \dots, i_m\}$. Chaque utilisateur a une liste d'items qu'il a noté. La liste des notes données par les utilisateurs à un ensemble d'items est représentée par une matrice $R: U \times I$. En général, les utilisateurs ne notent qu'un petit ensemble des items qui ont réellement consultés (regardés, lus, …). Par conséquent, on trouve beaucoup de valeurs manquantes dans la matrice R.

Le tableau 2.1 illustre un exemple de matrice d'évaluation concernant 5 utilisateurs et 8 items. Chaque utilisateur évalue une liste d'items pour exprimer son intérêt pour chaque article. Le but d'un système de recommandation est de prédire la note manquante dans la matrice et de recommander un item à un utilisateur si sa note prédite est élevée.

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8
u_1	2	4	4			5	1	
u_2		3	4	2			3	2
u_3	1	5				4		
u_4			3		3		4	
u_5		2	2	3	4		3	3

Tableau 2.1 : *matrice utilisateur x item*

Dans le chapitre état de l'art (section 1.2.2.2), nous avons discuté deux méthodes de filtrage collaboratif : les méthodes basées mémoire et les méthodes basées modèle. Les méthodes basées sur la mémoire fonctionnent sur l'ensemble de la matrice *R* et génèrent des recommandations en identifiant le voisinage de l'utilisateur candidat auquel la recommandation sera faite, en fonction des notes passées de l'utilisateur. Les techniques basées sur des modèles utilisent les notes des items pour former un modèle, puis le modèle sera utilisé pour dériver les recommandations.

Les techniques basées sur la mémoire sont assez efficaces dans les applications du monde réel car elles sont faciles à mettre en œuvre. Cependant, certains problèmes limitent leur application, en particulier dans les applications à grande échelle.

En effet, les notes manquantes dans la matrice d'évaluation et le fait qu'elle soit creuse représente un phénomène duquel le filtrage collaboratif souffre qui est la parcimonie – rareté des données (sparsity) –, où les utilisateurs n'évaluent qu'un petit ensemble d'items. Cela rend le calcul de la similarité entre les utilisateurs imprécis et réduit par conséquent la précision des algorithmes de FC (Bobadilla et al., 2013; Falih, 2018).

De plus, il est difficile de considérer dans le calcul les nouveaux utilisateurs qui n'ont pas encore noté aucun item ainsi que les nouveaux items qui n'ont pas encore été noté, il s'agit du problème de démarrage à froid.

Pour surmonter les faiblesses des techniques basées sur la mémoire, un axe de recherche s'est concentré sur les techniques de clustering notamment la détection de communautés basées sur des modèles dans le but de rechercher des méthodes plus efficaces (Pham et al., 2011). Basées sur les notations, ces techniques regroupent les utilisateurs ou les items en clusters offrant ainsi une nouvelle façon d'identifier le voisinage.

Dans notre travail, nous nous intéressons à l'utilisation des méthodes de détection de communautés afin de pouvoir trouver le voisinage d'un utilisateur (item).

En effet, la détection de communautés peut être vue comme une généralisation du principe du filtrage collaboratif où le principe est de recommander à une personne les items les mieux évalués par les membres de sa communauté. Les items peuvent être aussi regroupés en communautés selon leurs notes voire le motif de leur visualisation (achat, …). La matrice utilisateur-item R peut être vue comme une matrice d'adjacence d'un graphe biparti. Ce graphe biparti comprend l'ensemble des utilisateurs et l'ensemble des items. Ils sont liés par des notes qui représentent le niveau d'appréciation d'un item par un utilisateur.

Notre objectif consiste alors, à trouver, dans une première étape, les communautés des utilisateurs (items) similaires en se basant sur les techniques de détection de communautés. La seconde étape consiste à prédire les notes manquantes afin de recommander des items aux utilisateurs, ceci se fait en se basant sur les communautés appropriées au lieu de calculer les prédictions sur tout le graphe.

Dans ce qui suit, nous allons présenter en détails notre approche à travers une architecture composée de trois étapes.

2.3. Modèle proposé

Le modèle de recommandation que nous proposons est basé sur deux étapes principales, chacune de ces étapes est composée de sous-étapes.

Dans la première, en se basant sur la matrice d'évaluation $R: U \times I$ et en utilisant les techniques de détection de communautés, nous allons identifier des communautés d'utilisateurs et des communautés d'items (figure 2.1).

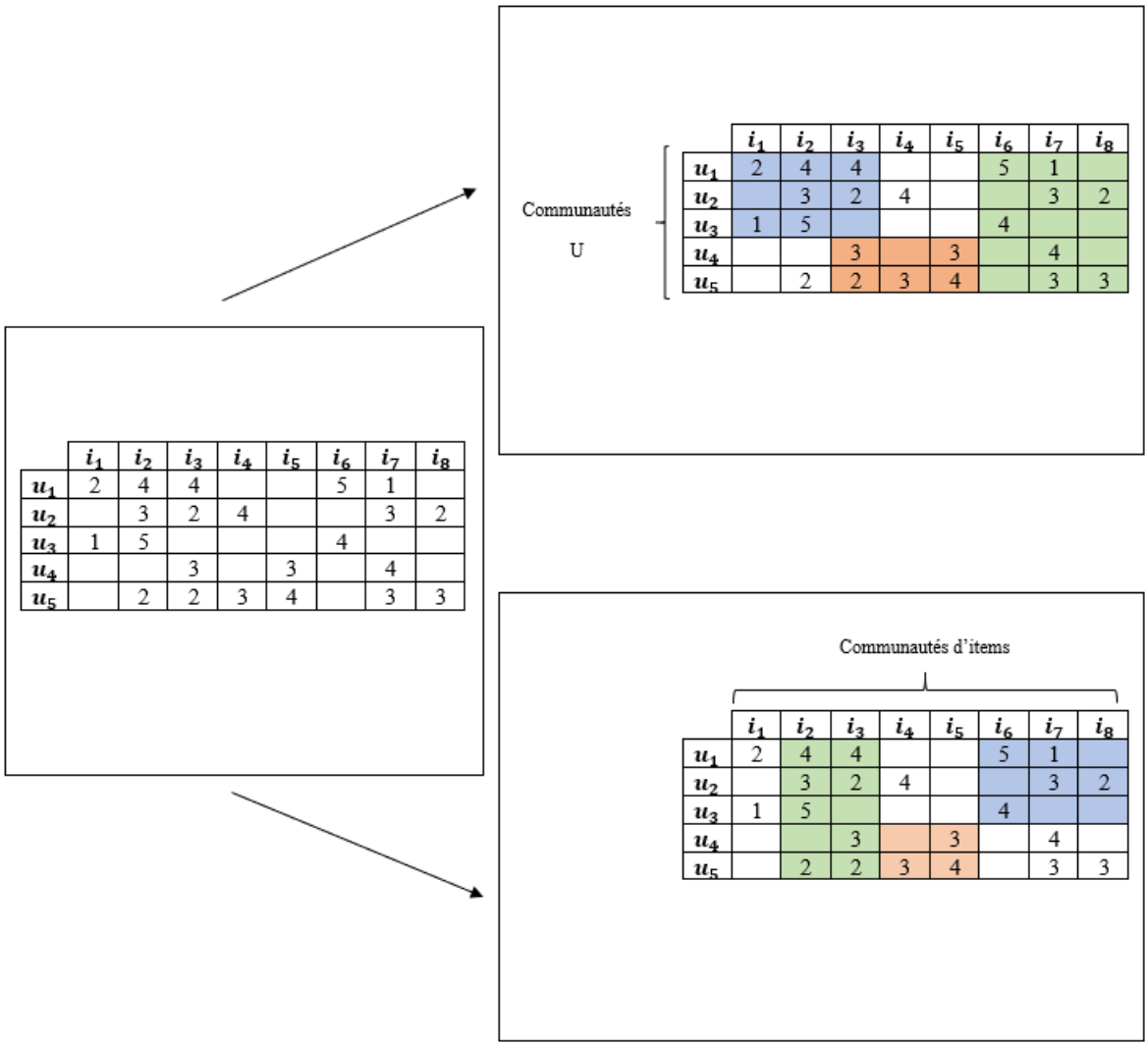


Figure 2.1 : *Les communautés utilisateurs – items.*

La deuxième étape reçoit comme entrée la sortie de l'étape précédente (communautés d'utilisateurs et d'items). A ce niveau, nous allons calculer la prédiction des notes pour identifier la note qu'un utilisateur u pourra donner à un item i et ainsi recommander aux utilisateurs les items avec des notes prédites élevées.

Dans ce qui suit, nous allons détailler les deux grandes étapes sus-citées ainsi que leurs sous-étapes à travers une architecture du modèle proposé.

2.3.1. Architecture proposée

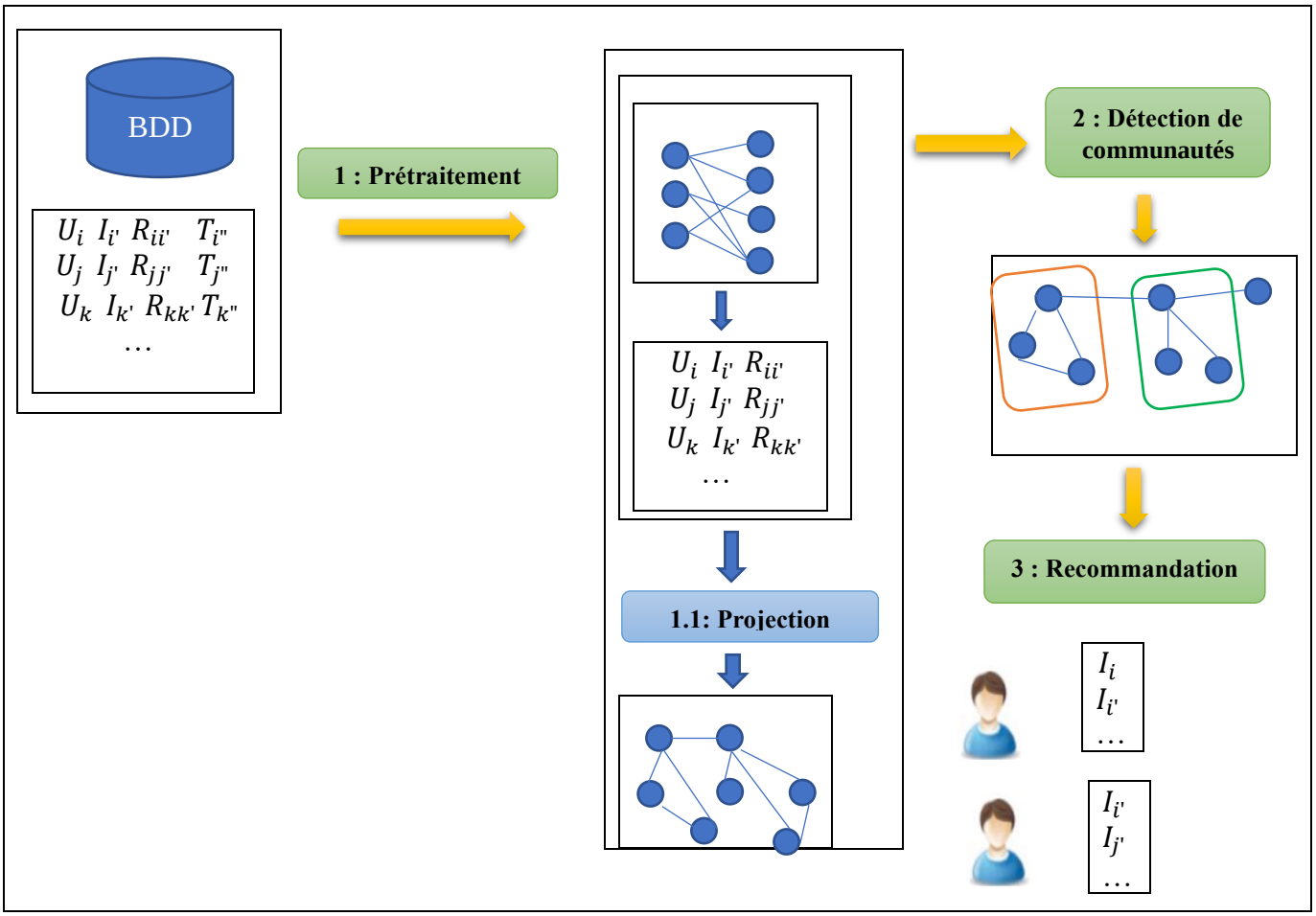


Figure 2.2 : *Architecture du système de recommandation proposé.*

2.3.1.1. Prétraitement

Dans cette étape, nous allons construire un graphe biparti à partir de la matrice des notes $R: U \times I$.

Tout d'abord, nous commençons par effectuer des pré-traitements sur la base de données (BDD) que nous utilisons. Cette BDD est représentée par une matrice utilisateur/item. Notre objectif

consiste à obtenir à partir de cette matrice $U \times I$, une matrice d'adjacence d'un graphe biparti où on a deux types de nœuds, à savoir, les utilisateurs et les items.

Les détails de la BDD utilisée et les pré-traitements effectuées seront donnés dans le prochain chapitre.

Une fois on obtient le graphe biparti, et afin de pouvoir appliquer les algorithmes de détection de communautés, nous allons à présent, faire une projection une fois sur les items et une fois sur les utilisateurs. Le résultat de la projection sera un graphe mono-parti avec un seul type de nœud (soit des items, soit des utilisateurs). De plus, nous tirons profit des notes données par les utilisateurs afin de générer des poids pour les liaisons entre les nœuds.

Les algorithmes de projection que nous proposons sont donnés ci-dessous.

2.3.1.1.1. Projection

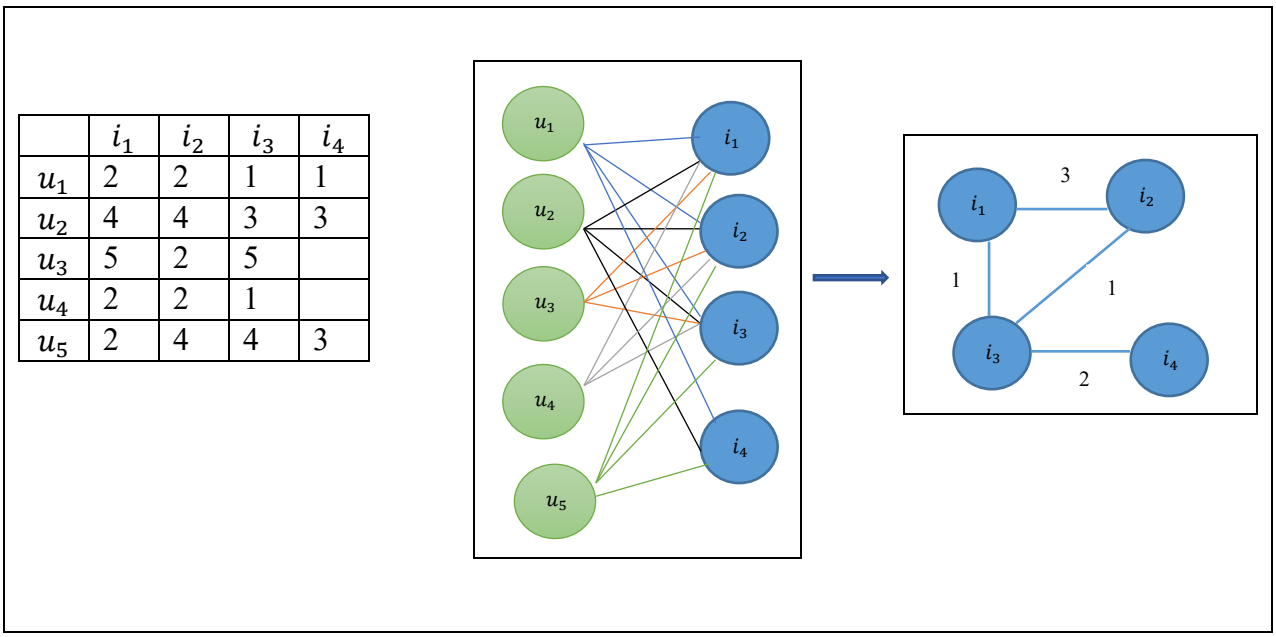
- *Algorithme de projection sur items*

Pour construire un graphe mono-parti pondéré avec comme type de nœuds les items, nous allons suivre les étapes suivantes :

1) **Deux items seront connectés s'ils ont été notés avec la même note par le même utilisateur.**
2) **Un poids est donné à chaque arête qui consiste au nombre de fois les items ont été notés avec la même note par différents utilisateurs.**

Exemple

La figure 2.3 présente un exemple de matrice $U \times I$, le graphe biparti correspondant et sa projection :



	i_1	i_2	i_3	i_4
u_1	2	2	1	1
u_2	4	4	3	3
u_3	5	2	5	
u_4	2	2	1	
u_5	2	4	4	3

Figure 2.3 : *Exemple de projection sur les items.*

Dans ce qui suit, nous allons donner le pseudocode de l'algorithme de projection sur les items.

Entrées : matrice d'évaluation – matrice d'adjacence de G biparti $[u_i\ i_j\ r_{i,j}]$
L_k- lignes de la matrice
Sortie : matrice d'adjacence de G mono-parti
Poids p

Début
$k = 1, p = 0$
1. **Pour tout** ($L \in Lignes\_matrice$) **faire**
2. **Si** $r_{L_k} = r_{L_{k+1}}$ **alors**
3. **Si** $u_{L_k} = u_{L_{k+1}}$ **alors**
4. Générer une liaison entre i_{L_k} et $i_{L_{k+1}}$
5. **Si** $i_{L_k} = i_{L_{k+1}}$ **alors**
6. **Si** $r_{L_k} = r_{L_{k+1}}$ **alors**
7. **Si** $u_{L_k} = u_{L_{k+1}}$ **alors**
8. $p = p + 1$
9. $k = k + 1$
10. **Fin si**
11. **Fin si**
12. **Fin si**
13. $p = 0$
14. **Fin si**
15. **Fin pour**
16. **Fin**

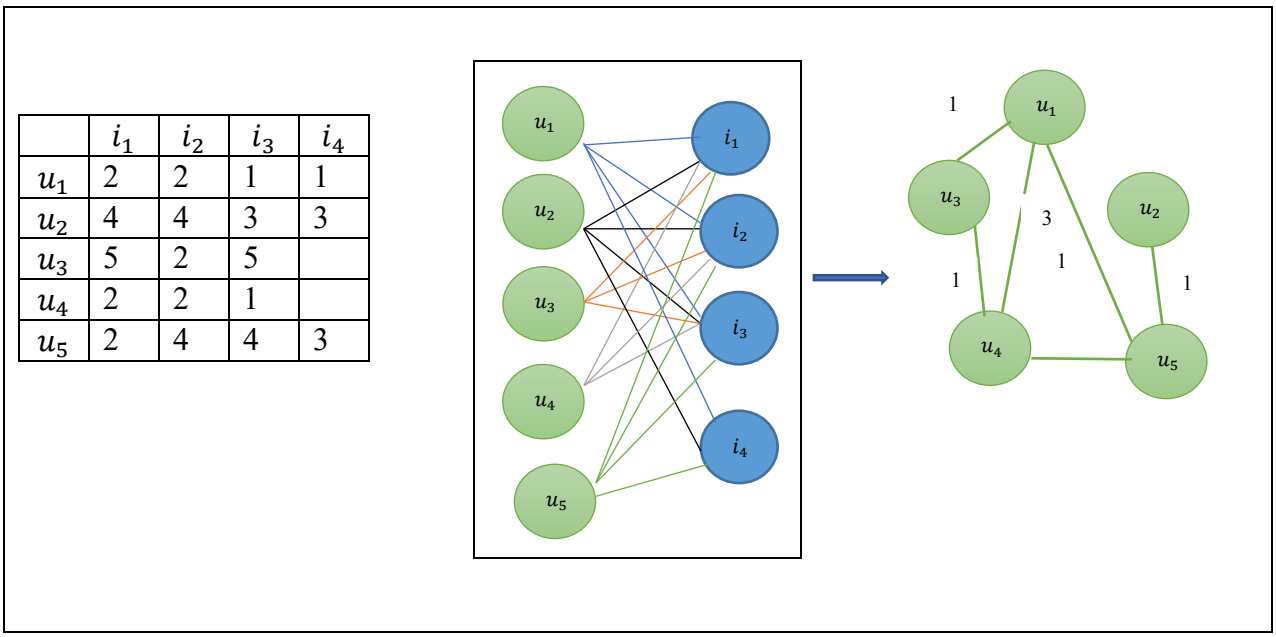
- *Algorithme de projection sur les utilisateurs*

Pour construire un graphe mono-parti pondéré avec comme type de nœuds les utilisateurs, nous allons suivre les étapes suivantes :

1) **Deux utilisateurs seront connectés s'ils ont donné la même note au même item.**
2) **Un poids est donné à chaque arête qui consiste au nombre de fois les utilisateurs ont donné la même note aux différents items**

Exemple

La figure 2.4 présente un exemple de matrice $U \times I$, le graphe biparti correspondant et sa projection :



	i_1	i_2	i_3	i_4
u_1	2	2	1	1
u_2	4	4	3	3
u_3	5	2	5	
u_4	2	2	1	
u_5	2	4	4	3

Figure 2.4 : *Exemple de projection sur les utilisateurs*

Dans ce qui suit, nous donnons le pseudocode de l'algorithme de projection sur les utilisateurs.

Entrées : matrice d'évaluation – matrice d'adjacence de G biparti $[u_i\ i_j\ r_{i,j}]$
L_k- lignes de la matrice
Sortie : matrice d'adjacence de G mono-parti
Poids p
Début
$k = 1, p = 0$

1. **Pour tout** ($L \in Lignes\_matrice$) **faire**
2. **Si** $r_{L_k} = r_{L_{k+1}}$ **alors**
3. **Si** $i_{L_k} = i_{L_{k+1}}$ **alors**
4. Générer une liaison entre u_{L_k} et $u_{L_{k+1}}$
5. **Si** $u_{L_k} = u_{L_{k+1}}$ **alors**
6. **Si** $r_{L_k} = r_{L_{k+1}}$ **alors**
7. **Si** $i_{L_k} = i_{L_{k+1}}$ **alors**
8. $p = p + 1$
9. $k = k + 1$
10. **Fin si**
11. **Fin si**
12. **Fin si**
13. $p = 0$
14. **Fin si**
15. **Fin pour**
16. **Fin**

2.3.1.2. Détection de communautés

Une fois que le graphe (graphe mono-parti après projection) soit construit dans l'étape de prétraitement, nous pouvons maintenant l'utiliser comme entrée dans la phase de détection de communauté.

Dans notre travail nous utilisons l'algorithme infomap (Rosvall & Bergstrom, 2008) pour détecter les communautés. Cet algorithme permet de détecter des communautés dans de différents types de graphes. Nous nous intéressons dans notre travail aux communautés disjointes dans des graphes statiques. En outre il a prouvé sa performance par rapport aux graphes pondérés (avec poids).

Il a été démontré dans de divers travaux à l'image de (Alzahrani & Horadam, 2016; Bedi & Sharma, 2016; El-moussaoui et al., 2019) que cet algorithme est très rapide avec une complexité linéaire (Tableau 1.1). (Alzahrani & Horadam, 2016) ont conclu dans leur travail qu'infomap dépasse 11 autres algorithmes expérimentés suivi par l'algorithme de Louvain (Blondel et al., 2008).

Notre choix est allé vers l'algorithme infomap car ce dernier a donné de meilleurs résultats en ce qui concerne les graphes pondérés (Alzahrani & Horadam, 2016).

- *Fonctionnement de l'algorithme infomap*

l'algorithme infomap proposé par (Rosvall & Bergstrom, 2008) exploite la compression des données, en compressant le mouvement d'un marcheur aléatoire sur un graphe. Il repose sur le principe : trouver une structure dans des graphes équivaut à résoudre un problème de codage.

En d'autres termes, si nous voulons comprendre comment la structure du graphe est liée au comportement du système, nous devons comprendre le flux d'information sur le graphe. Un groupe de nœuds parmi lesquels les informations circulent rapidement et facilement peuvent être agrégés en une seule communauté (nœuds fortement connectés).

Cet algorithme associe à chaque nœud un code en deux parties : un préfixe correspondant à la communauté ainsi qu'un identifiant du nœud dans la communauté. Un chemin peut être décrit par les identifiants des nœuds qu'il parcourt, et le préfixe de communautés traversées. L'objectif est de trouver les préfixes communautaires permettant de minimiser le nombre moyen de bits nécessaires pour représenter une marche aléatoire.
Cette optimisation est faite de manière gloutonne, puis raffinée par le biais d'un algorithme de recuit simulé.

Sa fonction objective est appelée « map equation », elle est basée sur le flux d'information et est utilisée pour trouver une représentation compressée d'un ensemble de marches aléatoires dans le graphe (Zeng & Yu, 2018). Elle est définie comme suit :

$$L(C) = q_{\curvearrowleft} H(Q) + \sum_{c \in C} p_{\circlearrowleft}^{c} H(P^{c}) \quad (2.1)$$

- C : l'ensemble des communautés
- $q_{\curvearrowleft} H(Q)$: les mouvements entre les communautés (q : la somme de la probabilité 'exit' de chaque communauté dans le graphe (la probabilité qu'un marcheur aléatoire change les communautés, H(Q) : la longueur moyenne du code des mouvements entre les communautés).
- $\sum_{c \in C} p_{\circlearrowleft}^{c} H(P^{c})$: les mouvements dans les communautés (p : la probabilité qu'un marcheur aléatoire reste dans une communauté, H(P) : la longueur moyenne du code du codebook d'une communauté (l'entropie des mouvements dans une communauté)).
- L (C) : représente la limite inférieure de la longueur de code de la structure communautaire détectée C.

L'algorithme infomap utilise la longueur de description minimale (MDL) (Zeng & Yu, 2018), pour mesurer la qualité des résultats communautaires détectés. A chaque itération, l'algorithme minimise le changement en MDL lors du déplacement d'un nœud isolé dans une communauté.

Cet algorithme est constitué de trois phases principales (Zeng & Hongfeng, 2018) :

Phase 1 : Calculer la probabilité de visite de chaque nœuds u (degré du nœuds sur l'ensemble du nœuds) :

$$p_u = degree(u)/|E| \quad (2.2)$$

- u : un nœud du graphe
- E : l'ensemble des arrêtes

Phase 2 : il suit une stratégie de clustering hiérarchique où chaque nœud est considéré comme une communauté unique.

- L'algorithme calcule la probabilité de visite de chaque communauté en utilisant la probabilité de chaque nœud :

$$\begin{gathered} pour\ tout\ u\ \in V^k\ faire: \\ C_u^k = \{u\} \\ p^{C_u^k} = \sum_{\propto \in C_u^k} p_\propto \end{gathered} \tag{2.3}$$

- L'algorithme calcule la probabilité de sortie de chaque communauté :

$$q^{C_u^k} = \sum w_{u,v}\ (u,v) \in E^k, u \in C_u^k\ et\ v \notin C_u^k \tag{2.4}$$

- Il calcule le gain en MDL de chaque mouvement d'un nœud de sa communauté originale à chaque communauté voisine afin d'atteindre un minimum changement en MDL :

$$\begin{gathered} répéter: \\ pour\ tout\ u \in V^k\ faire: \\ si\ C_u'^k = argmin\left(\delta L_{C_u^k \to C_u'^k}\right) < 0\ alors \\ C_u^k = C_u^k - \{\mathrm{u}\}; C_u'^k = C_u'^k \cup \{u\} \\ p^{C_u^k} = \sum_{\propto \in C_u^k} p_\propto - p_u; p^{C_u'^k} = \sum_{\propto \in C_u'^k} p_\propto + p_u \end{gathered} \tag{2.5}$$

$$mise\ à\ jour\ de\quad p^{C_u^k}\ et\ q^{C_u'^k} \tag{2.6}$$

- A la fin, l'algorithme calcule la nouvelle longueur MDL en utilisant les mises à jour p et q.

Phase 3 : l'algorithme fusionne les communautés en un nouveau graphe, où chaque nœud dans l'ensemble des nœuds représente une communauté dans l'ensemble des communautés courantes et chaque arrête dans l'ensemble des arrêtes représente toutes les arêtes qui relient les communautés. L'algorithme continue jusqu'à ce que les communautés deviennent stables (Zeng & Yu, 2018).

2.3.1.3. La recommandation

Une fois que les communautés soient construites dans l'étape précédente (détection de communauté) nous pouvons les utiliser comme entrée dans la partie de recommandation. En effet,

les communautés trouvées vont être exploitées afin d'aider le système de recommandation dans le calcul de la prédiction des notes des utilisateurs.

L'avantage de cette approche est que nous nous basons, lors du calcul, sur les items appartenant aux communautés, tandis que l'approche traditionnelle du filtrage collaboratif prend en compte tous les items de l'ensemble de données.

Notre objectif est de calculer la note qu'un utilisateur u peut donner à un item candidat i en se basant sur la liste des items appartenant à la même communauté que l'item i. A la fin la liste de recommandation pour l'utilisateur courant contient les items candidats qui ont une note prédite élevée.

Afin de trouver la liste de recommandation pour chaque utilisateur, nous suivons les étapes décrites ci-dessous :

Etape 1 : Pour chaque utilisateur, on identifie un *item cible,* l'item qui a la note la plus élevée.

Etape 2 : Les items qui ne sont pas notés par l'utilisateur courant et qui appartiennent à la communauté de l'item cible sont considérés comme items candidats[1].

Etape 3 : La similarité est calculée entre chaque item candidat et les items notés par l'utilisateur courant appartenant à la même communauté. Nous utilisons dans notre travail la similarité *cosine* donnée par l'équation (1.5) dans le chapitre précédent.

Etape 4 : Le calcul de la prédiction se fait à cette étape. Nous allons adapter la formule générale de calcul de prédiction en filtrage collaboratif (chapitre 1, équation (1.6)), pour prendre en considération les communautés. La formule de prédiction utilisée est donnée par l'équation (2.7).

Etape 5 : Pour chaque utilisateur, considérer les communautés des items qu'il a déjà noté en vue de calcul de prédiction.

Etape 6 : Générer la liste de recommandation. En se basant sur les notes prédites, recommander à l'utilisateur les items qui ont une note prédite élevée.

[1] Les items candidats se sont les items qui peuvent être recommandés à l'utilisateur courant.

La prédiction est calculée par :

$$P(u,i) = \frac{\sum_{j \in C} sim(i,j)\ r_{u,j}}{\sum_{j \in C} |sim(i,j)|} \quad (2.7)$$

Où :

u est l'utilisateur courant (Pour lequel on va calculer la prédiction et ainsi la recommandation).

i est l'item candidat pour l'utilisateur u (qu'on va prédire sa note).

C est la communauté des items à laquelle l'item i appartient.

$r_{u,j}$ est la note donnée par l'utilisateur u à l'item j.

$sim(i,j)$ est la similarité entre les deux items i et j.

2.4. Conclusion

Dans ce chapitre, nous avons présenté les différentes étapes méthodologiques par lesquelles passent notre travail. Nous avons donné l'architecture de notre système et expliqué par la suite chacune de ses phases et sous phases jusqu'à ce qu'on atteigne notre objectif, à savoir le calcul de la recommandation en se basant sur des communautés.

Dans le chapitre suivant, nous allons donner les détails d'implémentation de notre travail.

Chapitre 3 : Implémentation

3.1. Introduction

Nous décrivons dans ce chapitre, les travaux applicatifs ainsi que les détails d'implémentation réalisés pendant ce travail. Nous allons décrire dans un premier temps la base de données utilisée pour valider notre approche. Nous décrivons, dans un second temps, les outils et le langage d'implémentation. Par la suite, nous détaillons l'implémentation des étapes de notre travail et nous terminons par les résultats obtenus et les expérimentations.

3.2. Base de données

3.2.1. Description de la base

Dans notre travail, nous avons utilisé la base de données des films MovieLens du groupe *GroupLens Research Center* (Resnick & Varian, 1997) afin de tester la performance du système de recommandation proposé. Cette base de données est téléchargeable à partir du site MovieLens[2]. Elle est largement utilisée par la communauté scientifique pour évaluer les algorithmes de recommandation. Elle a l'avantage d'être créée sur des notes réelles et ainsi elle fournit un bon support de validation.

La base de données MovieLens est constituée de 100,000 notes données sur une échelle de 1 à 5 (R) par 943 utilisateurs (U) sur 1682 (I) films. Chaque utilisateur a noté au moins 20 films, ce qui fait que cette matrice montre 93,70% de notes manquantes (Falih, 2018). Cette base est partagée en 80% des données en une base d'apprentissage et 20% en une base de test.

3.2.2. Echantillon de données

Pour effectuer nos expérimentations, nous avons choisi un échantillon de données de la base de données MovieLens.

[2] https://grouplens.org/datasets/movielens/

Afin d'assurer une bonne diversité dans les catégories de films regardés et notés par les utilisateurs, nous avons effectué un échantillonnage basé sur de différentes tranches d'âge des utilisateurs. Les tranches d'âge étudiées sont données dans le tableau 3.1, deux âges de chaque tranche ont été choisi, et par la suite deux utilisateurs de chaque âge ont été choisi de chaque tranche :

$\hat{a}ge < 20$
$20 \leq \hat{a}ge < 30$
$30 \leq \hat{a}ge < 40$
$40 \leq \hat{a}ge < 50$
$\hat{a}ge \geq 50$

Tableau 3.1 : *les différentes tranches d'âge des utilisateurs traités.*

Les utilisateurs retenus sont donnés dans le tableau suivant :

Identifiant	**Age**	**Sexe**	**Fonction**	**Code postal**
6	42	M	executive	98101
12	28	F	Other	6405
34	38	F	administrator	42141
52	18	F	Student	55105
54	22	M	executive	66315
56	25	M	librarian	46260
77	30	M	technician	29379
90	60	M	educator	78155
91	55	M	marketing	1913
101	15	M	Student	5146
126	28	F	Lawyer	20015
138	46	M	Doctor	53211
142	13	M	Other	48118
149	35	F	marketing	17325
151	38	F	administrator	48103
195	42	M	scientist	93555
220	30	M	librarian	78205

340	46	M	engineer	80123
377	22	M	student	18015
413	55	M	educator	78212
584	25	M	student	27511
609	13	F	student	55106
618	15	F	student	44212
787	18	F	student	98620
882	35	M	engineer	40503
900	60	M	retired	18505

Tableau 3.2 : *Les utilisateurs de notre échantillon de données.*

Il faut noter qu'on n'a pas fait de restrictions sur les items notés par ces utilisateurs-là, tous les items qui ont été noté, ont été retenu pour notre échantillon de données.

3.2.3. Exploration des données

Comme nous venons de l'exposer la matrice $R: U \times I$ de la base de données contient peu de notes par rapport aux nombres d'utilisateurs et d'items, on dit que la matrice est creuse. Ce phénomène est très connu dans les systèmes de recommandation à savoir la parcimonie – rareté des données ou encore en anglais *sparsity.*

La figure 3.1 montre la rareté des données dans la base de données movielens [2].

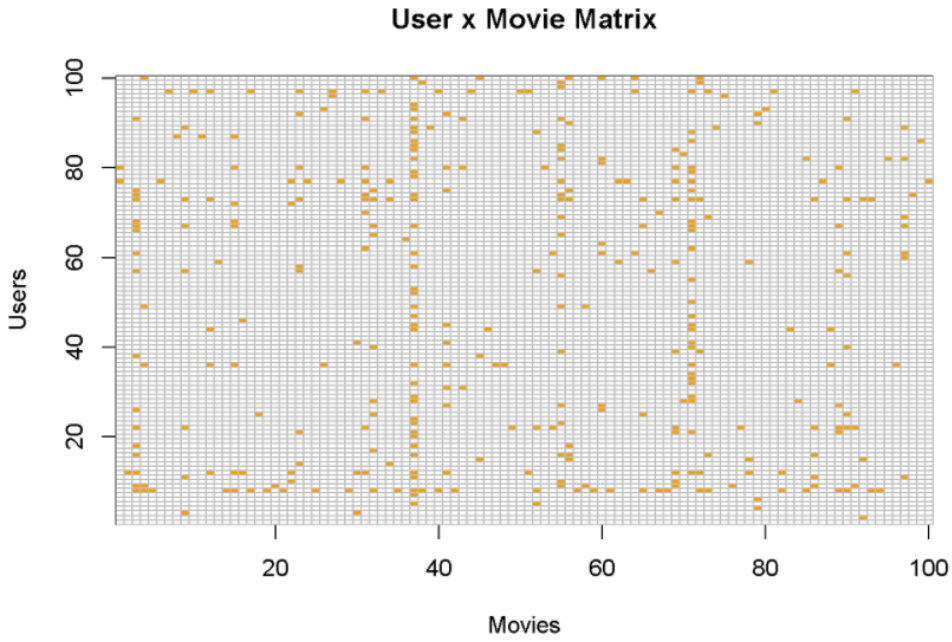


Figure 3.1 : *l'effet de sparsity dans la base de données MovieLens.*

Dans la figure 3.2 on montre la rareté de données dans l'échantillon des données choisi. On remarque que beaucoup de films n'ont pas été noté.

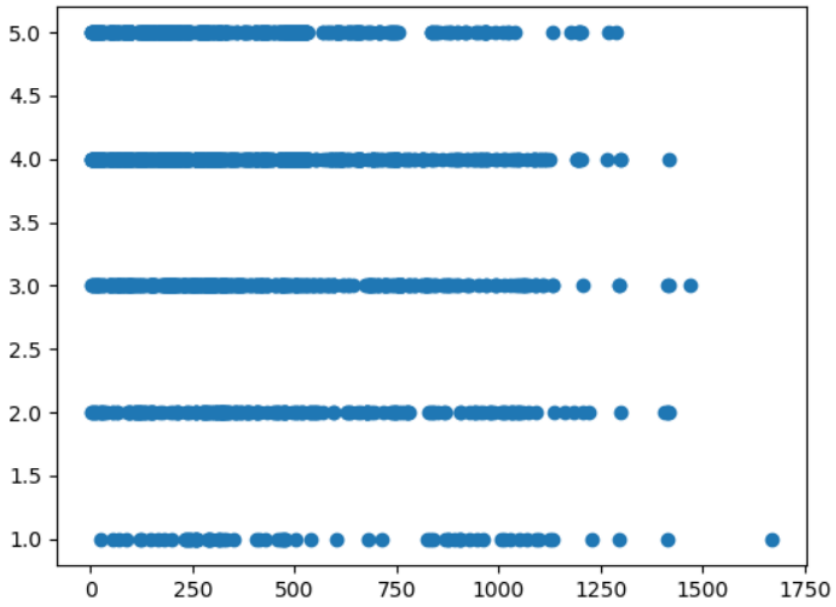


Figure 3.2 : *l'effet de sparsity et la distribution des notes dans notre échantillon de données.*

3.3. Mise en œuvre de la contribution

Dans cette partie, nous allons spécifier les outils utilisés pour développer notre application.

3.3.1. Outils et langage

- *Le langage Python*

Nous avons choisi le langage python comme langage de programmation. Il a été créé en 1989 par Guido van Rossum (Fuchs & Pierre, 2020). C'est un langage puissant et facile à apprendre. Il a des structures de données de haut niveau, efficaces et une approche simple mais efficace de la programmation orientée objet (Rossum, G, 2018). Il fonctionne sur de nombreux système d'exploitation : Windows, linux, Android, et iO, etc., avec une syntaxe élégante et un typage dynamique (Fuchs & Pierre, 2020).

C'est un langage interprété ce qui fait que c'est un langage idéal pour la création de scripts et le développement rapide d'application dans de nombreux domaine sur diverses plates-formes (Rossum, G, 2018)**.**

Il est très utilisé dans le cadre d'analyse de données, analyse des réseaux sociaux et la bio-informatique (Fuchs & Pierre, 2020)

- *L'IDE Pycharm*

PyCharm (figure 3.3) est un environnement de développement intégré utilisé pour programmer en Python.

Il permet l'analyse de code et contient un débogueur graphique. Il permet également la gestion des tests unitaires, l'intégration de logiciel de gestion de versions, et supporte le développement web avec Django. Développé par l'entreprise tchèque JetBrains, c'est un logiciel multi-plateforme qui fonctionne sous Windows, Mac OS X et Linux. Il est décliné en édition professionnelle, diffusé sous licence propriétaire, et en édition communautaire diffusé sous licence Apache [3].

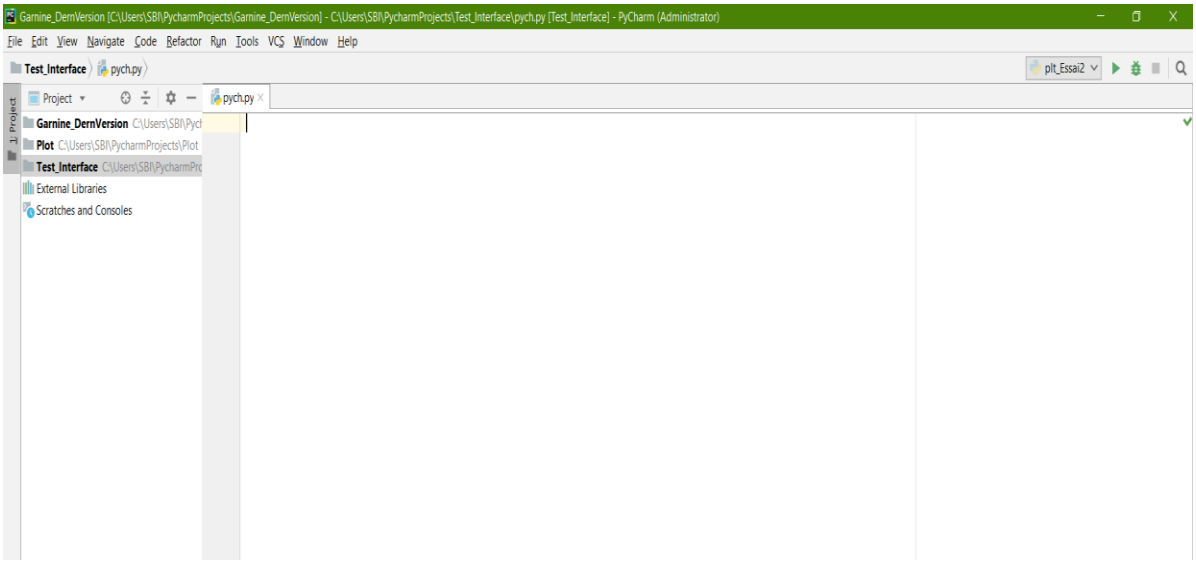


Figure 3.3 : *L'environnement Pycharm.*

- *Package Igraph*

Igraph est une collection de bibliothèques pour créer et manipuler des graphes et analyser des réseaux. Il est écrit en C et existe également sous forme de packages Python et R. Le logiciel est largement utilisé dans la recherche universitaire en science des réseaux et dans les domaines connexes. La publication qui présente le logiciel compte 5623 citations au 5 juin 2015 selon Google Scholar [4].

3.3.2. Description du système et résultats

Dans cette section, nous présentons le système de recommandation réalisé à travers quelques interfaces graphiques et la sortie de chaque étape du travail.

L'interface principale de notre système est donnée par la figure 3.4.

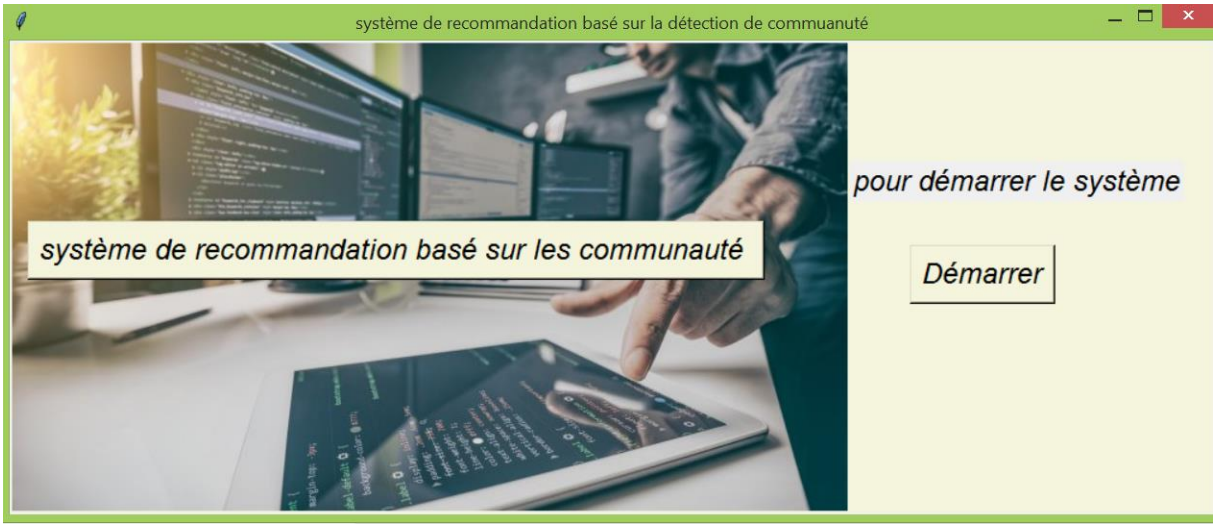


Figure 3.4 *: Interface principale du système.*

3.3.2.1. Prétraitement

La base de données MovieLens est une base de données temporelles. Avec les notes des items données par les utilisateurs, on trouve le temps de l'attribution de note en seconde. En effet, dans notre travail nous nous intéressons aux graphes statiques. Cet aspect de temps nous mène vers une dynamique du graphe que nous ne gérons pas.

De ce fait, la première étape à faire dans le prétraitement de données est de ne garder que les données qui sont importantes pour la réalisation de notre système.

Les données qui nous intéressent sont les trois métriques de la matrice R, à savoir, l'ensemble des utilisateurs U, l'ensemble des items I et l'ensemble des notes r (figure 3.5). L'utilisateur de notre système peut charger le fichier de la base de données, comme il peut charger le fichier que nous utilisons après prétraitement (figure 3.6).

90	1193	4
90	1194	4
90	1195	5
90	1196	4
90	1197	4
90	1198	5
90	1199	5
90	1200	4
90	1204	4
90	1205	3
90	1206	2
91	22	5
91	28	4
91	31	5
91	50	5
91	56	1
91	64	4
91	79	5
91	82	5
91	97	5
91	98	5
91	99	2
91	127	5

Figure 3.5 : *Un fragment de la base de données* $\langle u, i, r_{u,i} \rangle$

Figure 3.6 : *Interface relative aux chargements des fichiers de base de données.*

- *La projection du graphe sur les items*

Après avoir implémenté l'algorithme de projection proposé dans le chapitre précédant, nous avons obtenu un graphe mono-parti avec un seul type de nœuds, à savoir, les items, et des arêtes pondérées avec poids qui est relatifs au nombre de fois où deux items ont été notés par la même note de la part de différents utilisateurs (figure 3.7 et 3.8).

Figure 3.7 : *Interface relative au graphe généré après projection.*

```
les items projetés sont : 23 536
 le poids est :  1
les items projetés sont : 23 537
 le poids est :  1
les items projetés sont : 28 95
 le poids est :  1
les items projetés sont : 28 111
 le poids est :  2
les items projetés sont : 28 117
 le poids est :  2
les items projetés sont : 28 143
 le poids est :  4
les items projetés sont : 28 237
 le poids est :  3
les items projetés sont : 28 257
 le poids est :  1
les items projetés sont : 28 258
 le poids est :  1
les items projetés sont : 28 276
 le poids est :  1
les items projetés sont : 28 284
 le poids est :  1
les items projetés sont : 28 286
 le poids est :  1
les items projetés sont : 28 294
 le poids est :  1
```

Figure 3.8 : *Les liaisons entre les items avec leurs poids.*

3.3.2.2. Détection de communautés

Après avoir obtenu la projection des items dans l'étape précédente, nous avons besoin d'un fichier .gml spécifique au package igraph comme entrée pour cette étape afin de pouvoir appliquer l'algorithme infomap et obtenir les communautés.

La figure suivante montre une partie de notre fichier gml :

```
    id 1671
]
node [
    id 1116
]
node [
    id 101
]
node [
    id 1412
]
node [
    id 696
]
node [
    id 744
]
node [
    id 129
]
edge [
    source 1
    target 8
    value 1
]
edge [
    source 1
    target 12
    value 3
]
edge [
    source 1
    target 19
    value 1
]
edge [
    source 1
    target 23
    value 1
```

Figure 3.9 : *un fragement du fichier .gml généré par notre programme.*

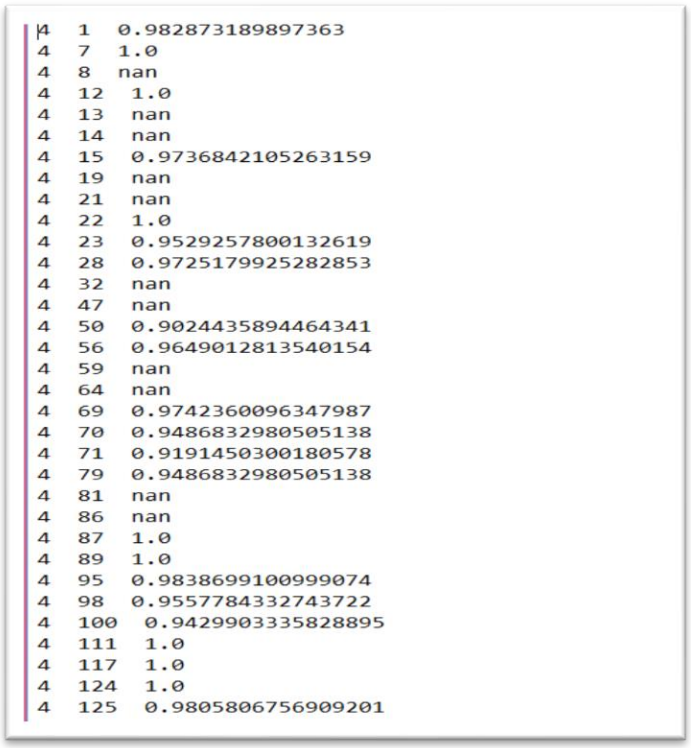
Les communautés résultantes sont données à la figure 3.10.

```
 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 4, 4, 4, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 6, 6, 6, 6, 6, 6, 6, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 8, 0,
0, 0, 0, 0, 4, 0, 0, 0, 9, 0, 0, 0, 0, 0, 0, 0, 10, 0, 10, 1, 0, 0, 0, 0, 3, 11, 12, 3, 12, 12, 13, 5, 12
7, 12, 13, 12, 12, 4, 13, 4, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 13, 0, 13, 13, 0, 13, 13, 13,
0, 0, 13, 13, 13, 13, 0, 13, 13, 13, 13, 13, 0, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 0, 13, 13
13, 13, 11, 5, 11, 11, 7, 11, 11, 11, 11, 0, 0, 2, 0, 0, 0, 0, 0, 0, 12, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 2, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 2, 0, 0, 4, 4, 10, 4, 5, 4, 4, 4, 4, 0, 7, 7, 7, 7, 7, 5, 7, 7,
7, 11, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 10, 5, 6, 0, 14, 14, 14, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0,
0, 0, 0, 0, 8, 0, 8, 8, 8, 8, 8, 8, 9, 9, 9, 9, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
5, 5, 5, 5, 5, 5, 5, 5, 15, 15, 15, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 16, 16, 1, 1, 1, 1, 1, 1, 0, 0, 0, 10,
10, 10, 10, 10, 10, 10, 10, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 6, 6, 3, 0, 0, 0, 0, 0, 0]
```

Figure 3.10 : *Les communautés détectées des items.*

3.3.2.3. La recommandation

Dans la recommandation, nous avons besoin du résultat de l'étape précédente c'est-à-dire les communautés comme entrée de cette étape. Nous avons utilisé les items et leurs communautés pour calculer la similarité (figure 3.11) entre les items de la même communauté. Après le calcul de similarité, nous avons calculé la prédiction de note pour chaque utilisateur et chaque item qui n'a pas été noté par l'utilisateur.



```
4  1   0.982873189897363
4  7   1.0
4  8   nan
4  12   1.0
4  13   nan
4  14   nan
4  15   0.9736842105263159
4  19   nan
4  21   nan
4  22   1.0
4  23   0.9529257800132619
4  28   0.9725179925282853
4  32   nan
4  47   nan
4  50   0.9024435894464341
4  56   0.9649012813540154
4  59   nan
4  64   nan
4  69   0.9742360096347987
4  70   0.9486832980505138
4  71   0.9191450300180578
4  79   0.9486832980505138
4  81   nan
4  86   nan
4  87   1.0
4  89   1.0
4  95   0.9838699100999074
4  98   0.9557784332743722
4  100   0.9429903335828895
4  111   1.0
4  117   1.0
4  124   1.0
4  125   0.9805806756909201
```

Figure 3.11 : *Résultat du calcul de la similarité.*

Pour un utilisateur courant, après avoir calculé la similarité et la prédiction nous générons la liste des items que notre système va lui recommander. Dans cette liste on ne garde que les items dont leur note prédite est égale ou supérieur à 3.0.

La figure 3.12 montre le résultat de la recommandation pour l'utilisateur 6 à titre d'exemple.

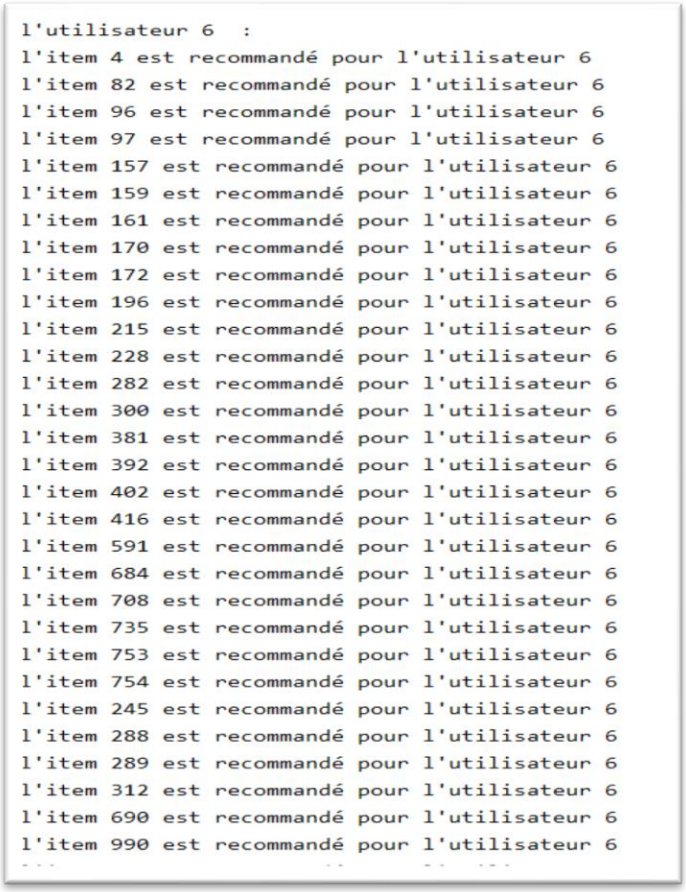


Figure 3.12 : *Résultat de recommandation pour l'utilisateur 6.*

3.3.2.4. Expérimentation

Nous allons, à présent, évaluer le système de recommandation proposé en calculant les deux métriques présentées dans le chapitre 1 (section 1.2.3), L'erreur MAE (équation 1.7) et l'erreur RMSE (équation 1.8). Nous calculons ces deux erreurs pour l'utilisateur 6 à titre d'exemple.

Le tableau 3.3 présente un échantillon de la note réelle que l'utilisateur 6 donne à quelques items tirés de la base de test et la note calculée (prédite) par notre système pour les mêmes items.

	Note réelle	Note prédite
Utilisateur 6	4	3
	4	3
	2	3
	5	3
	4	3
	4	4
	3	3

Tableau 3.3 : *comparaison entre la note réelle de quelques items et la note prédite par notre système.*

Les deux métriques MAE et RMSE calculées pour évaluer les notes prédites pour l'utilisateur 6 sont données dans le tableau 3.4.

	MAE	**RMSE**
Utilisateur 6	**0.857**	**1.069**

Tableau 3.4 : *Evaluation du système de recommandation.*

Il est clair que la marge d'erreur de notre système est acceptable. Cela nous permet de conclure que l'architecture que nous avons proposée qui est une combinaison entre le filtrage collaboratif et la détection de communautés donnent de résultats prometteurs.

3.4. Conclusion

Dans ce chapitre, nous avons présenté notre base de données, les outils de développement de ce système, les étapes d'implémentation. Nous avons présenté également les résultats de chaque étape de notre travail ainsi que les résultats d'évaluation de notre système.

Les résultats obtenus sur notre échantillon de données sont encourageants, mais ils nécessitent encore d'autres évaluation en utilisant d'autres métriques.

Conclusion générale

Les systèmes de recommandation sont une forme spécifique de filtrage d'information visant à présenter les éléments d'information qui sont susceptibles d'intéresser l'utilisateur. Il existe plusieurs techniques de recommandation, nous nous sommes intéressées dans ce mémoire à la technique de filtrage collaboratif qui est basée sur une matrice d'évaluation utilisateur x item.

Malgré que cette technique représente la méthode la plus utilisée dans le cadre de recommandation, elle souffre de la rareté des données. En effet, la matrice utilisateur x item est en général incomplète et contient plusieurs données manquantes. Ceci affecte la qualité et la précision de la recommandation.

Pour pallier à ce problème, nous avons proposé d'utiliser les techniques de détection de communautés afin de pouvoir trouver le voisinage d'un utilisateur/item. L'avantage de notre approche est que nous nous basons, lors du calcul, sur les items appartenant aux communautés, tandis que l'approche traditionnelle du filtrage collaboratif prend en compte tous les items de l'ensemble de données. Les résultats que nous avons obtenus à travers à cette étude sont encourageants et valident notre idée initiale de bénéficier de la structure graphique et des techniques de détection de communautés afin de combler le problème de parcimonie.

Des perspectives d'amélioration de notre travail restent, toutefois, indispensables.

A cours terme, nous envisageons, dans un premier temps, d'appliquer notre approche sur la totalité de la BDD MovieLens afin de tester le passage à l'échelle de notre approche. Dans un second temps, nous envisageons d'implémenter la technique classique du filtrage collaboratif (Baseline) afin de pouvoir comparer nos résultats.

Nos perspectives à long terme concernent les autres problèmes des SR, principalement le démarrage à froid. Pour pallier à ce problème notre intérêt reste le même pour les techniques de détection de communautés mais nous envisageons de générer d'autres graphes en se basant sur les informations des utilisateurs et des items afin d'agréger les communautés et injecter d'autres informations aux graphes.

Références bibliographiques

Adomavicius, G., & Tuzhilin, A. (2005). 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*, *17*(6), 734–749.

Agarwal, G., & Kempe, D. (2008). Modularity maximizing graph communities via mathematical programming. *The European Physical Journal B-Condensed Matter and Complex Systems*, *66*(409–418).

Aldecoa, R., & Marín, I. (2013). Surprise maximization reveals the community structure of complex networks. *Scientific Reports*, *January*. https://doi.org/10.1038/srep01060

Alzahrani, T., & Horadam, K. J. (2016). *Community Detection in Bipartite Networks: Algorithms and Case studies. Complex Systems and Networks*, Springer, Vol. 325, Issue 5939, pp. 406–406. https://doi.org/10.1007/978-3-662-47824-0

Arenas, A., Albert, D., & Conrad, J. P. (2008). Synchronization reveals topological scales in complex networks. *Phys Rev Lett*, 1–4.

Bedi, P., & Sharma, C. (2016). *Community detection in social networks*. *Data Mining Knowledge Discovery*. *6*(June), 115–135. https://doi.org/10.1002/widm.1178

Belloui, A. (2008). *L'usage des concepts du web sémantique dans le filtrage d'information collaboratif*. Thèse de Doctorat, Ecole supérieur d'informatique, Alger.

Berry, J. W., Bruce, R. A., Hendrickson, & Phillips, C. (2009). Tolerating the community detection resolution limit with edge weighting. *Physical Review*.

Blondel, V. D., Guillaume, J., Lambiotte, R., & Lefebvre, E. (2008). *Fast unfolding of communities in large networks*. *Journal of statistical Mechanics : Theory and Experiment* 1–12.

Bobadilla, J., Ortega, F., Hernando, A., & Gutiérrez, A. (2013). Recommender Systems Survey. *Knowledge-Based Systems*, *46*, 109–132.

Brandes, U., Gaertler, M., & Wagner, D. (2009). *Experiments on Graph Clustering Algorithms*. *Proceedings of the 11th European Symposium on Algorithms, Lncs 2832*, 568–579.

Burke, R. (2002). Hybrid Recommender Systems : survey and experiments. *User Modeling and User-Adapted Interaction*, *12*(4), 331–370.

Cai, Q., & Lijia, M. (2016). A survey on network community detection based on evolutionary computation A survey on network community detection based on evolutionary computation. *International Journal of Bio-Inspired Computation*. https://doi.org/10.1504/IJBIC.2016.076329

Canu, M. (2017). *Détection de communautés orientée sommet pour des réseaux mobiles opportunistes sociaux*. Thèse de Doctorat, Université Pierre et Marie Curie.

Chakrabarti, D. (2004). Autopart : parameter-free graph partioning and outlier detection. *Lecture Notes in Computer Science*, 112–124.

Clauset, A., Newman, M. E. J., & Moore, C. (2004). Finding community structure in very large networks. *Physical Review*.

Combe, D. (2014). *Détection de communautés dans les réseaux d'information utilisant liens et attributs*. École doctorale "Sciences, Ingénierie'''."

Crawford, N., & Cho, S. (2015). *Community Detection & Network Analysis for Beer Recommendations*. Stanford Project Proposal, 1–6.

Creusefond, J. (2016). *Caractériser et détecter les communautés dans les réseaux sociaux*. Thèse de Doctorat, Université de Caen Normandie.

El-moussaoui, M., Agouti, T., Tikniouine, A., & El Adnani, M. (2019). A comprehensive literature review on community detection : Approaches and applications. *Procedia Computer Science*, *151*, 295–302. https://doi.org/10.1016/j.procs.2019.04.042

Falih, I. (2018). *Attributed Network Clustering : Application to recommender systems*. Thèse de Doctorat, Université Sorbonne, Paris.

Falih, I., Grozavu, N., Kanawati, R., & Bennani, Y. (2018). Topological multi-view clustering for collaborative filtering. *Procedia Computer Science*, *144*, 306–312. https://doi.org/10.1016/j.procs.2018.10.524

Fortunato, S, Latrova, V., & Marchiori, M. (2004). Method to find community structures based on

information centrality. *Physical Review*.

Fortunato, Santo. (2010). Community detection in graphs. *Physics Reports*, *486*(3–5), 75–174. https://doi.org/10.1016/j.physrep.2009.11.002

Fortunato, Santo. (2016). Community detection in networks: A user guide. *Physics Reports*, 1–43.

Fuchs, P., & Pierre, P. (2020). *Cours de Python*. Notes de Cours.

Gasparetti, F., Micarelli, A., & Sansonetti, G. (2018). Community Detection and Recommender Systems. *Encyclopedia of Social Network Analysis and Mining - 2nd Edition*.

Girvan, M., & Newman, M. E. J. (2002). Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 1–8.

Gondron, M., & Minoux, M. (1995). *Graphes et algorithmes* (Eyrolles (ed.); Eyrolles). Eyrolles. https://www.eyrolles.com/Informatique/Livre/graphes-et-algorithmes-9782212015713/

Gorripati, S. K., & Vatsavayi, V. K. (2017a). A community based content recommender systems. *International Journal of Applied Engineering Research*, *12*(22), 12989–12996.

Gorripati, S. K., & Vatsavayi, V. K. (2017b). Community-Based Collaborative Filtering to Alleviate the Cold-Start and Sparsity Problems. *International Journal of Applied Engineering Research*, *12*(15), 5022–5030.

Gregory, S. (2009). *Finding overlapping communities in networks by label propagation*. *New Journal of Physics*, 12(10), 103018.

Hamadache, B. (2017). *Analyse et recherche dans les réseaux sociaux: Vers la caractérisation et l'identification significative d'une identité de structure noyau possible au sein d'un processus évolutionnaire décrivant la dynamique d'un réseau social*. Thèse de Doctorat, Université Badji Mokhtar Annaba.

Hastings, M. (2006). Community detction as an inference problem. *Physical Review*.

Isinkaye, F. O., Folajimi, Y. O., & Ojokoh, B., A. (2015). *Recommendation Systems : Principles methods and evaluation*. *Egyptian Informatics Journal*, 16, 261-273.

Kamahara, J., Asakawa, T., Shimojo, S., & Miyahara, H. (2005). A community-based

recommendation system to reveal unexpected interests. *Proceedings of the 11th International Multimedia Modelling Conference, MMM 2005*, 433–438. https://doi.org/10.1109/MMMC.2005.5

Kanawati, R. (2013). *Détection de communautés dans les grands graphes d'interactions (multiplexes) : état de l ' art*.

Khan, B. S., & Niazi, M. A. (2016). *Network Community Detection : A Review and Visual Survey*. *arXiv preprint arXiv:1708.00977*.

Kumar, R., Novak, J., & Tomkins, A. (2006). Structure and evolution of online social networks. *12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 611–617.

Lancichinetti, A., & Fortunato, S. (2009). *Community detection algorithms: a comparative analysis*. *Physical Review E,* 80(5), 1–12.

Leung, I. X. Y., Hui, P., & Li, P. (2009). Towards Real-Time Community Detection in Large Networks. *Physical Review*.

Linden, G., Smith, B., & York, J. (2003). Amazon.com Recommendations : item-to-item collaborative filtering. *IEEE Internet Computing*, *7*(1), 76–80.

Meghanathan, N. (n.d.). *Community Detection Algorithms*. *Algorithms*, 9(1).

Mkhitaryan, K. K. (2019). Realization of Recommender Framework Based on Community Detection. *Mathematical Problems of Computer Science*, *51*, 57–65.

Müler, D. (2012). Introduction à la théorie des graphes. *Cahiers de La CRM*, *6*, 52.

Nagwekar, K., & Shirsat, P. K. (2017). A Community Detection and Recommendation System. *Ijarcce*, *6*(1), 7–13. https://doi.org/10.17148/ijarcce.2017.6102

Nedioui, M. A. (2015). *Fouille et apprentissage automatique dans les réseaux sociaux dynamiques*. Mémoire de Magister, Université de Biskra.

Nettleton, D. F. (2013). Data mining of social networks represented as graphs. *Computer Science Review*, *7*, 1–34.

Newman, M. E. . (2004a). Detecting community structure in networks. *The European Physical Journal*, *330*, 321–330. https://doi.org/10.1140/epjb/e2004-00124-y

Newman, M. E. J. (2003). The structure and function of complex networks. *SIAM Review*, *45*(2), 167–256. https://doi.org/10.1137/S003614450342480

Newman, M. E. J. (2004b). *Fast algorithm for detecting community structure in networks*. *Physical Review E*, 69(6).

Newman, M. E. J. (2016). Finding community structure in networks using the eigenvectors of matrices. *Physical Review*, *74*(3).

Papadopoulos, S., Kompatsiaris, Y., Vakali, A., & Spyridonos, P. (2012). Community detection in Social Media Performance and application considerations. *Data Mining Knowledge Discovery*, *24*, 515–554. https://doi.org/10.1007/s10618-011-0224-z

Papagelis, M., Plexousakis, D., & Kutsuras, T. (2005). Alleviating the sparsity problem of collaborative filtering using trust inferences. *International Conference on Trust Management*, 224–239.

Parthasarathy, S., Ruan, Y., & Satuluri, V. (2011). *Community Discovery in Social Networks : Applications, Methods and Emerging Trends*. In *Social network data analytics* (pp. 79-113). Springer, Boston, MA. https://doi.org/10.1007/978-1-4419-8462-3

Pham, M. C., Cao, Y., Klamma, R., & Jarke, M. (2011). *A clustering approach for collaborative filtering recommendation using social network analysis*. *J. UCS*, *17*(4), 583-604.

Radicchi, F., Castellano, C., Cecconi, F., Loreto, V., Parisi, D., & Fisica, D. (2004). *Defining and identifying communities in networks*. *Proceedings of the national academy of sciences*, *101*(9), 2658-2663.

Raghavan, U. N., & Kumara, S. (2007). Near linear time algorithm to detect community structures in large-scale networks. *Physical Review*, 1–12.

Rao, N., & Talwar, V. (2008). Application domain and functional classification of recommender systems : a survey. *Desidoc Journal of Library and Information Technology*, *28*(3), 17–36.

Rashid, A. M., Albert, I., Cosley, D., Lam, S. K., McNee, S. M., Konstan, J. A., & Riedl, J. (2002).

Getting to know you : learning new user preferences in recommender systems. *7th International Conference on Intelligent User Interfaces IUI*, 127–134.

Reichardt, J., & Bornholdt, S. (2006). Statistical mechanics of community detection. *Physical Review*.

Resnick, P., & Varian, H. R. (1997). Recommender Systems. *Communication of the ACM*, *40*, 56–58.

Rossum, G, V. (2018). *Python tutorial*. Notes de cours.

Rosvall, M., & Bergstrom, C. (2008). Maps of random walks on complex networks reveal community structure. *Proc Natl Acad Sci*, *105*, 1118–1123.

Sahebi, S., & Cohen, W. W. (2011). Community-Based Recommendations : a Solution to the Cold Start Problem. *Workshop on Recommender Systems and the Social Web (RSWEB)*.

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). *Item-based collaborative filtering recommendation algorithms*. In *Proceedings of the 10th international conference on World Wide Web* (pp. 285-295).

Sawant, S. (2013). Collaborative Filtering using Weighted BiPartite Graph Projection - A Recommendation System for Yelp. *Proceedings of the CS224W: Social and Information Network Analysis Conference*. http://web.stanford.edu/class/cs224w/projects2013/cs224w-038-final.pdf

Schaeffer, S. E. (2007). Graph clustering. *Computer Science Review*, *1*, 27–64. https://doi.org/10.1016/j.cosrev.2007.05.001

Scott. (2000). *Social network analysis : A handbook.* 2nd edn sage publications.

Sigward, E. (2002). *Introduction à la théorie des graphes*. *Académie Metz-Nancy*.

Traag, V. A., Krings, G., & Dooren, P. Van. (2013). Significant Scales in Community Structure. *Scientific reports*, *3*(1), 1-10.

Van Dongen, S. (2000). *Graph clustering by flow simulation*. Thèse de Doctorat, Dutch National Research Institute for Mathematics and Computer Science, Utrecht, Netherlands.

Vragovic, I., & Louis, E. (2006). Network community structure and loop coefficient method. *Physical Review*.

Xiang, J. Z. Z., Wang, H. J., Li, Y., Zhang, S., Chan, C. C., & Guo, L. J. (2017). Comparing local modularity optimization for detecting communities in networks. *International Journal of Modern Physics*, *28*(6).

Xie, J., Kelly, S., & Szymanski, B. (2013). *Overlapping Community Detection in Networks : the State of the Art and Comparative Study*. *Acm computing surveys (csur)*, *45*(4), 1-35.

Xin, L., E, H., Song, J., Song, M., & Tong, J. (2014). Book recommendation based on community detection. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *8351 LNCS*, 364–373. https://doi.org/10.1007/978-3-319-09265-2_37

Yang, J., & Leskovec, J. (2012). Defining and Evaluating Network Communities based on Ground-truth. *Knowledge and Information Systems*, *42*(1), 181–213.

Zeng, J., & Yu, H. (2018). A Distributed Infomap Algorithm for Scalable and High-Quality Community Detection. *ICPP 2018: 47th International Conference on Parallel Processing*.

Zhao, G., Lee, M. L., Hsu, W., Chen, W., & Hu, H. (2013). Community-based user recommendation in uni-directional social networks. *International Conference on Information and Knowledge Management, Proceedings*, 189–198. https://doi.org/10.1145/2505515.2505533

Ziani, A. (2018). *La recommandation via l'analyse d'opinions*. Thèse de Doctorat, Université Badji Mokhtar, Annaba.

Webographie

[1] http://eric.univ-lyon2.fr/~ricco/cours/slides/WM.C%20-%20Detection%20de%20communautes.pdf (Mars 2019)

[2] https://rpubs.com/vsi/movielens (consulté le 20/09/2020)

[3] https://fr.wikipedia.org/wiki/PyCharm (consulté le 20/09/2020)

[4] https://en.wikipedia.org/wiki/Igraph (consulté le 20/09/2020)