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 : Sciences et Technologies de l'information et de la Communication

Thème :

Une approche basée sur la prédiction de liens pour les systèmes de recommandation

Encadré Par :

Dr. Abdelmoumène Hiba

Présenté par :

Rehailia Riane

Septembre 2021

Remerciement

Je remercie dieu pour m'avoir donné santé, courage et patience afin de m'aider à réaliser ce travail.

Mes remerciements s'adressent à mon encadrante, l'enseignante « Dr. Abdelmoumène Hiba *», pour avoir accepté de diriger ce travail, c'est un honneur pour moi.*

Pour ses multiples conseils et les efforts pour assurer un travail de qualité, son soutien, ses compétences et sa clairvoyance m'ont été d'une aide inestimable.

Je tiens à remercier sincèrement les membres du jury pour l'intérêt qu'ils ont porté à notre projet en acceptant d'examiner le travail et de l'enrichir par leurs propositions.

Enfin, je remercie tous ceux qui de près ou de loin ont contribué à l'accomplissement de ce travail.

Dédicace

Je dédie ce travail A mes parents qui ont partagés avec moi tous les moments d'émotion lors de la réalisation de ce travail,

Mon père pour son encouragement et ma mère qu'elle était la base de mon existence et le bonheur infini, pour l'effort qu'elle suscité en moi.

A mes frères Mohemed Ghoulam Allah et Mehdi Mourthada pour leurs encouragements durant tout mon parcours.

Ma grand-mère, mes tantes, mes oncles, mes cousins qu'ils m'ont chaleureusement supportée et encouragée tout au long de mon parcours.
A mes proches et particulièrement Meriem, Wissame, Bouchra et Racha pour leur tendresse, leur complicité et leur présence malgré la distance qui nous sépare.

A tous mes amis qui m'ont toujours encouragé, et à qui je souhaite plus de succès.

Merci !

Riane

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 (appelées aussi 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.

Nous nous intéressons dans ce projet à l'exploration des approches topologiques pour le calcul de recommandation afin de pallier à certains problèmes des méthodes classiques. En effet, la matrice d'évaluation peut être vue comme une matrice d'adjacence d'un graphe biparti qui relie les deux ensembles utilisateurs et items. La problématique de recommandation se réduit alors à un problème de prédiction de liens dans un graphe biparti.

L'objectif de notre travail est de réaliser un système de recommandation basé sur les graphes où la première étape consiste à collecter et explorer des informations liées aux items afin de faciliter l'intégration d'un nouvel item. La deuxième étape consiste à exploiter les informations tirées de la première étape lors du calcul de prédiction de liens entre les utilisateurs et les items.

Mots clés : Graphe biparti, Prédiction de liens, Système de recommandation, Filtrage collaboratif.

Table des matières

Liste des figures

Liste des tableaux

Introduction générale

Les systèmes de recommandation (SR) sont un sujet de recherche populaire qui vise à aider les utilisateurs à trouver des articles qui pourront les intéresser en fournissant des suggestions qui correspondent étroitement à leurs intérêts.

De différents algorithmes de recommandation ont été appliqués pour fournir un mécanisme automatique et intelligent permettant de filtrer l'excès d'informations disponibles pour les utilisateurs et de faire des recommandations personnalisées d'informations, de produits et de services lors d'une interaction en direct.

Une des approches les plus efficaces pour créer des systèmes de recommandation est le filtrage collaboratif (*CF, Collaborative Filtering*). Elle utilise les préférences connues sous forme de notes d'un groupe d'utilisateurs sur un ensemble d'items pour prédire leurs préférences inconnues à d'autres items et ainsi les recommander. Cette technique ainsi définie et utilisée souffre de plusieurs limites qui sont dues à plusieurs causes. D'un côté, le manque d'information relative aux préférences des utilisateurs (peu d'utilisateurs expriment explicitement leurs préférences) peut réduire la qualité de recommandation. De l'autre côté, l'incapacité de gérer l'arrivée d'un nouvel utilisateur et/ou un nouvel item puisqu'il n'a pas encore d'historique de préférences.

Pour surmonter ces faiblesses, la recherche dans ce domaine s'est orientée vers l'utilisation des graphes qui peuvent être une source d'information et de relations latentes entre les utilisateurs et les items. En effet, les utilisateurs et les items ainsi que la relation entre eux peuvent être présentés par un graphe biparti et le problème de recommandation sera traduit par un problème de prédiction de liens dans ce graphe.

Nous nous intéressons, dans notre travail, à la prédiction de liens dans un graphe *utilisateur-item* en vue de la recommandation. Nous allons, dans un premier temps, explorer les corrélations existantes entre les différents items afin de construire un premier graphe mono-parti indépendant des utilisateurs et de leurs appréciations. Ceci va nous faciliter l'intégration d'un nouvel item comme il va nous fournir des informations utiles qui vont guider la phase de prédiction de liens.

Dans un second temps, nous allons adapter les techniques de prédiction de liens afin de prendre en compte les spécificités du graphe biparti.

Ce mémoire est organisé comme suit :

Chapitre 1 : nous présentons, dans ce chapitre, les concepts de base de la théorie des graphes ainsi que le problème de prédiction de liens dans les graphes et les différentes techniques pour le résoudre.

Chapitre 2 : nous décrivons, dans ce chapitre, les systèmes de recommandation, les différentes approches de recommandation ainsi qu'une synthèse des travaux qui ont utilisé la prédiction de liens dans la recommandation.

Chapitre 3 : 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 4 : nous présentons dans ce chapitre les détails d'implémentation de notre système. Nous discutons, par la suite, les résultats trouvés et les évaluations élaborées.

Chapitre 1 : Prédiction de liens dans les graphes

1.1. Introduction

La prédiction de liens est une tâche qui consiste à prédire des relations et des interactions dans un graphe. En raison de son importance, la tâche de prédiction de liens, a reçu une grande attention des chercheurs de diverses disciplines. Ainsi, un grand nombre de méthodes ont été proposées au cours des dernières années.

Dans ce chapitre, nous allons effectuer un état de l'art sur la prédiction de liens. Nous allons tout d'abord, donner une brève présentation des concepts de base de la théorie des graphes. Nous donnons par la suite le fondement théorique de la prédiction de liens ainsi qu'une classification de ses différentes techniques de résolution.

1.2. Concepts de base de la théorie des graphes

Nous allons présenter dans cette section les notions fondamentales relatives à la théorie des graphes ainsi que les différents types de ces derniers.

1.2.1. Définition d'un graphe

Un graphe est un ensemble de sommets (ou appelés nœuds) noté V (pour *Vertices*, en anglais) et d'arêtes notés E (pour *Edges*, en anglais) liant certains couples de nœuds. On écrit $G = (V, E)$ où $V = \{v_1, v_2, \dots, v_n\}$ est l'ensemble de nœuds et $E = \{e_1, e_2, \dots, e_m\}$ est l'ensemble d'arêtes.

L'arête $(v_i, v_j) \in E$ est dites incidente à v_i et v_j et ces nœuds sont appelés voisins ou adjacents. Un nœud v qui n'est adjacent à aucun autre nœud du graphe est appelé nœud isolé.

On parle d'arête lorsque le graphe est non orienté et d'arcs lorsque le graphe est orienté.

Un graphe est défini par son ordre qui représente son nombre de nœuds (Sigward 2002) , (Canu 2017) . La figure 1.1 présente un graphe d'ordre 4.

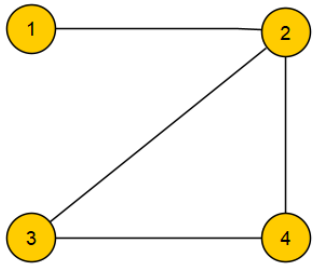


Figure 1.1 : *Exemple d'un graphe d'ordre 4.*

1.2.2. Propriétés des graphes

Nous présentons dans ce qui suit quelques propriétés des graphes.

- **Voisinage :** dans un graphe $G = (V, E)$, le voisinage d'un nœud v_i est noté $\Gamma(v_i)$, tel que : $\Gamma(v_i) = \{v_j \in V | (v_i, v_j) \in E\}$. Par exemple, les voisins du nœud 2 de la figure 1.1 ci-dessus sont les nœuds : 1, 3 et 4.
- **Le degré :** dans un graphe $G = (V, E)$ le degré $d(v)$ d'un nœud v est donné par le nombre de ses voisins. Le degré moyen d'un graphe, noté $\bar{d}$ est défini par (Sigward 2002) :

$$\bar{d} = \frac{1}{|V|} \sum_{i \in V} d(i)$$

  Par exemple, le degré du nœud 1 de la figure 1.1 est $d(1) = 1$.
- **Distance :** dans un graphe $G = (V, E)$, la distance entre deux nœuds v_i et v_j est la longueur du plus court chemin joignant ces deux nœuds, on note : $d(v_i, v_j)$.

  La distance entre nœud 1 et nœud 3 selon l'exemple dans la figure1.1 est :

  d (1,3) = 2
- **Diamètre :** le diamètre d'un graphe $G = (V, E)$ est le maximum des distances entre les nœuds de ce graphe, on note :

$$diam(G) = max\{d(v_i, v_j) | v_i, v_i \in V\}$$

  Le diamètre du graphe de la figure 1.1 est :

  diam(G) = 3

- **Densité :** c'est le rapport entre le nombre d'arêtes observées et le nombre maximal d'arêtes possibles. Une densité égale à 0 veut dire que tous les nœuds sont isolés et une densité égale à 1 veut dire qu'il existe un lien entre chaque paire de nœuds (graphe complet) (Matias 2018) :

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

La densité du graphe de la figure 1.1 est :

$$den(G) = \frac{2\times4}{1+(3\times2)} = 1{,}14$$

1.2.3. Définitions matricielles des graphes

Les graphes peuvent être définis par différentes matrices. Nous donnons ici la définition de la matrice d'adjacence et de la matrice de degrés.

1.2.3.1. Matrice d'adjacence

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

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

La matrice d'adjacence du graphe donné à la figure 1.1 est :

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

1.2.3.2. Matrice de degré

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). Soit $G = (V, E)$ et $v_i \in V$, la matrice de degrés $D_{i,j}$ est défini par :

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

La matrice des degrés D du graphe de la figure 1.1 est :

$$D = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}$$

1.2.4. Types des graphes

Nous allons donner dans cette section quelques types des graphes.

- **Sous-graphe :** c'est un graphe obtenu en supprimant certains nœuds et tous les arêtes liées aux nœuds supprimés, on a $H = (Y, B)$ est un sous-graphe de $G = (V, E)$ si $Y \subseteq V$ et $B \subseteq E$.
- **Graphe partiel :** c'est un graphe obtenu en supprimant certaines arêtes On a :

$H = (Y, B)$ est un graphe partiel de $G = (V, E)$ si $Y = V$ et $B \subseteq E$.

- **Graphe complet :** un graphe $G\ (V, E)$ est dit complet si deux nœuds quelconques distincts sont toujours adjacents.

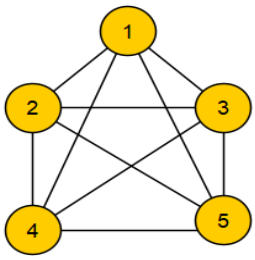


Figure 1.2 : *Exemple d'un graphe complet*

Le graphe dans la figure 1.2 ci-dessus représente un graphe complet d'ordre 5 car on a chaque nœud à une interaction avec tous les autres nœuds.

- **Graphe biparti :** un graphe biparti (ou bigraphe) est un réseau dont les nœuds sont divisés en deux ensembles disjoints X et Y et les seuls liens dans le graphique sont ceux qui connectent un nœud de X à un nœud de Y. Il est défini comme suit : $G = (X, Y, E)$ où X et Y sont deux ensembles de nœuds, E est un ensemble d'arêtes de G et est un sous-ensemble de $X \times Y$ (Benchettara, Kanawati et Rouveirol 2010).

  Un exemple d'un graphe biparti est donné dans la figure 1.2.

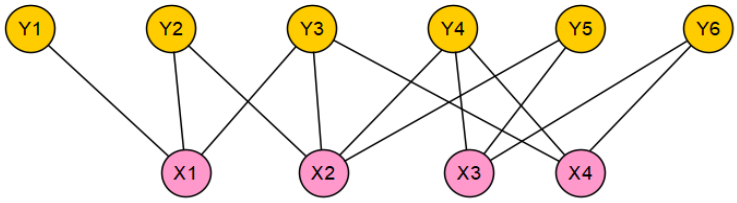


Figure 1.3 : *Exemple d'un graphe biparti.*

1.3. Prédiction de liens

La prédiction de liens est la tâche de prédire des relations dans un graphe ou réseau, son objectif est de trouver des liens manquants ou cachés (dans les graphes statiques) qui ne figurent pas dans le graphe actuel ou de prédire la probabilité d'apparition de liens futurs (dans les graphes dynamiques). En raison de sa large portée (la biologie, la physique, l'informatique et bien d'autres domaines), de nombreuses études se sont orientées vers ce domaine, et plusieurs méthodes ont été conçues et appliquées pour trouver et prédire des liens dans différents types de réseaux.

La plupart des travaux de la littérature sur la prédiction de liens se concentrent sur la tâche de l'existence de liens potentiels. En effet, la prédiction de liens peut viser plusieurs tâches. Elle peut être étendue à la prédiction de poids des liens, le type de la liaison ou encore la cardinalité du lien (Hasan et Svitlana 2014). La figure 1.4 présentent les différents objectifs que peut viser la prédiction de liens.

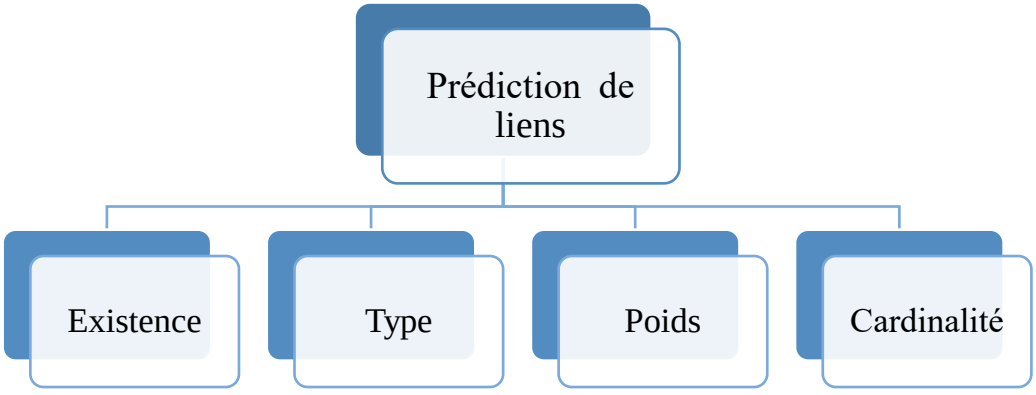


Figure 1.4 : *Les tâches de prédiction de liens* (Hasan et Svitlana 2014).

1.3.1. Domaines d'application de la prédiction de liens

Entre 1999 et 2020, des scientifiques en ingénierie, des informaticiens, des biochimistes, des spécialistes en télécommunication, des biologistes, des généticiens et des scientifiques de l'environnement se sont intéressés à la résolution du problème de la prédiction de liens et ainsi son application dans leurs différents domaines (figure 1.5). La figure 1.6 montre le nombre total d'articles publiés sur *Web of Science* dans le domaine de la prédiction de liens entre 1999 et 2020 (Wang et Le 2020).

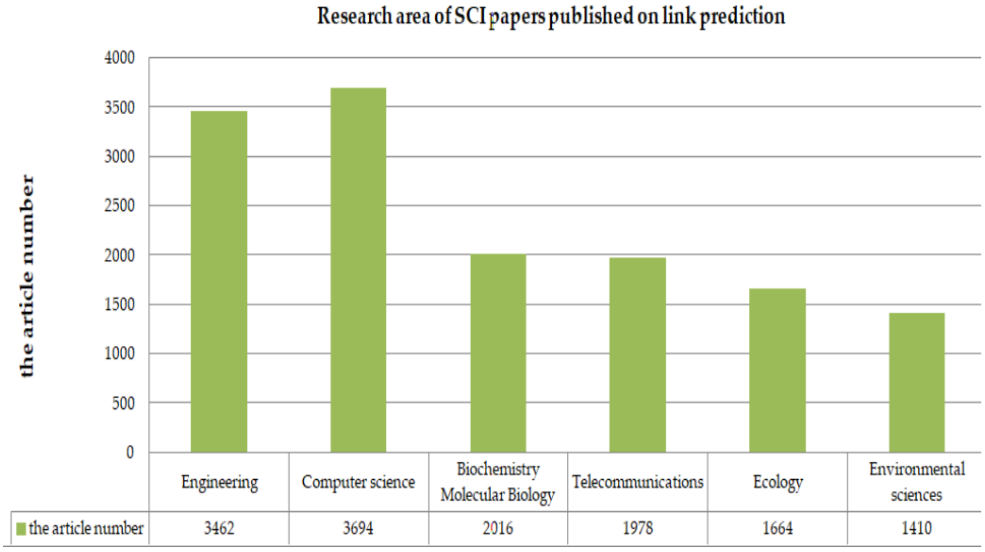


Figure 1.5 : *Les différentes disciplines qui ont abordées le problème de prédiction de liens* (Wang et Le 2020).

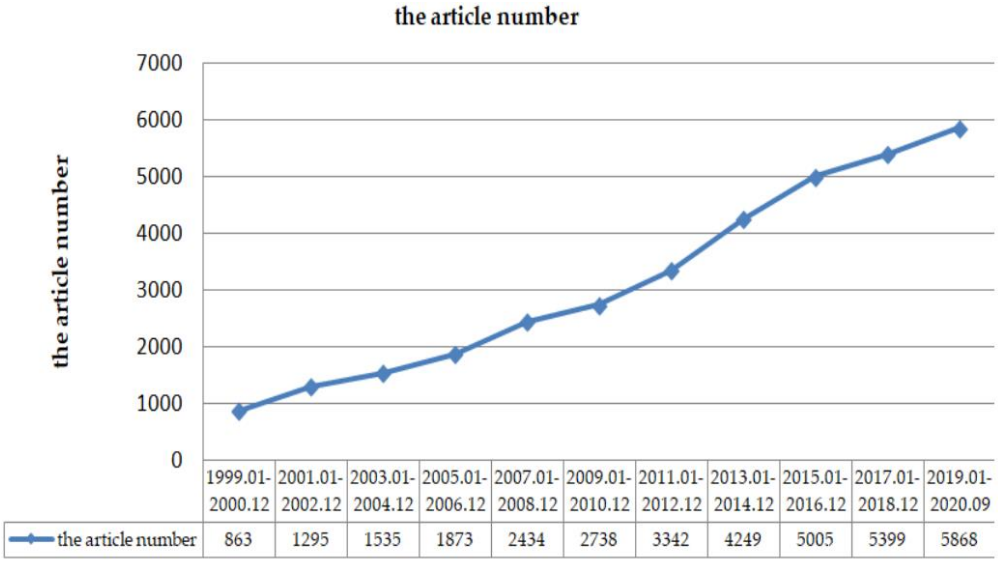


Figure 1.6 : *Le nombre d'articles scientifiques publiés sur la prédiction de liens*. (Wang et Le 2020)

Dans ces différentes disciplines, plusieurs problèmes ont été formalisés sous forme de prédiction de liens à l'image des réseaux sociaux où c'était appliqué pour prédire de nouvelles liaisons amicales, prédire le comportement des actions d'un utilisateurs, … ((Virinchi et Pabitra 2016); (Daud, et al. 2020); (Srilatha et Ramakrishnan 2016)) ; les réseaux biologiques pour prédire l'interaction Protéine-Protéine (PPI) (Kovács, et al. 2019), prédire l'interaction médicament-cible (Koptelov, Zimmermann et Crémilleux, et al. March 2020) ; les réseaux routiers (Bhuyan, Bhattacharyya et Kalita 2016) ; …

Les techniques de prédiction de liens ont été aussi largement utilisé dans les systèmes de recommandation dans plusieurs applications : e-commerce, e-tourisme, e-learning, recommandation des auteurs, ((Kaya 2019) ; (Li, et al. 2014) ; (Slokom et Ayachi 2018)).

1.3.2. Définition formelle de la prédiction de liens

On considère un graphe simple et non orienté $G = (V, E)$ où V est l'ensemble de nœuds et E est l'ensemble des arêtes. Une version complète de ce graphe, qu'on appelle U, contient un total de $\frac{|V|(|V|-1)}{2}$ liens. $(|U| - |E|)$ est le nombre de liens qui n'existent pas dans ce graphe mais qui peuvent apparaître dans le futur. Trouver ces liens manquants est l'objectif de la prédiction de liens (Ajay, et al. 2020).

Liben-Nowell et Kleinberg (Liben-Nowell et Kleinberg 2004) ont défini le problème de prédiction de liens tel qu'on suppose qu'on a un instantané du graphe G, noté $G_{t_0-t_1}(V,E)$ durant l'intervalle $[t_0, t_1]$ et $E_{t_0-t_1}$ est l'ensemble de liens présents dans cet instantané. La tâche de prédiction de liens consiste alors à trouver les liens $E_{t_0'-t_1'}$ durant l'intervalle $[t_0', t_1']$ où $[t_0, t_1] \leq [t_0', t_1']$.

Un exemple de la tâche de prédiction de liens est donnée dans la figure 1.7, où au temps t, quatre liaisons seulement sont présents dans le graphe et au temps t', avec $t' > t$, trois autres liens sont apparus (liens en pointillés).

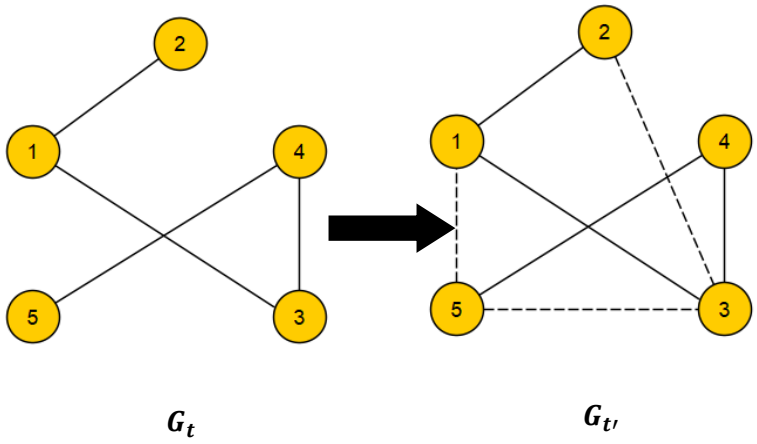


Figure 1.7 : *Prédire des liens dans un graphe.*

1.3.3. Classification des approches de prédiction de liens

Le domaine de la prédiction de liens est un domaine très étudié dans la littérature et a été servi par plusieurs surveys qui ont étudié les différentes techniques de prédiction et qui ont proposé différentes classifications de ces techniques ((Ajay, et al. 2020) ; (Wang et Le 2020); (Li, Fang et Sheng 2017)).

Nous adoptons dans notre travail la classification de (Li, Fang et Sheng 2017) où ils ont classé les approches de prédiction de liens dans deux grandes classes : les approches basées sur l'apprentissage et les approches basées sur la proximité. Dans chacune de ces grandes classes apparait un ensemble de familles de méthodes explorées afin de servir la prédiction de liens (figure 1.8).

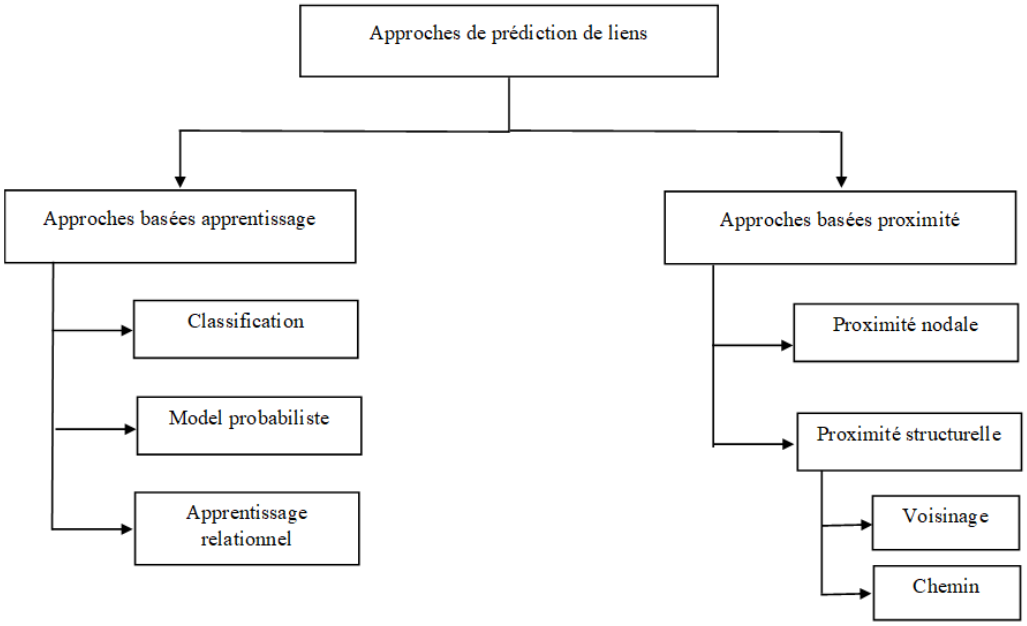


Figure 1.8 : *Classification des approches de prédiction de liens.*

1.3.3.1. Approches basées sur l'apprentissage

Les méthodes basées sur l'apprentissage apprennent un modèle à partir des données d'apprentissage construites à partir des établissements de liens observés et utilisent le modèle appris pour prédire la probabilité de liaison pour chaque lien potentiel. Un modèle peut être appris à l'aide des approches de classification, des modèles probabilistes ou des méthodes d'apprentissage relationnel.

1.3.3.1.1. Classification

Une majorité de méthodes de cette catégorie est un apprentissage supervisé. Après avoir identifié un ensemble de caractéristiques, le problème de prédiction de liens est mis en correspondance avec une classification binaire. Les chercheurs dans ce cadre ont utilisé : les machines à vecteurs de support (SVM) (GONG, TALWALKAR et MACKEY 2014), la régression logistique, (Wang, Satuluri et Parthasarathy 2007), les arbres de décision (Benchettara, Kanawati et Rouveirol 2010), et bien d'autres techniques.

1.3.3.1.2. Modèle probabiliste

Les méthodes basées sur des modèles probabilistes prédisent la probabilité de liaison entre nœuds. Le modèle probabiliste optimisera une fonction cible construite pour établir un modèle composé d'un groupe de paramètres, qui correspond le mieux aux données observées du graphe. Puis la probabilité de l'existence d'un lien inexistant est estimée par la probabilité conditionnelle. (Lu et Tao 2010). Le modèle de bloc stochastique a été largement utilisé dans ce cadre ((Guimerà et Sales-Pardo, Missing and spurious interactions and the reconstruction of complex networks 29, 2009) ; (Clauset, Moore et Newman 2008)). D'autres auteurs ont utilisé le coefficient de clustering (Z. Huang 2010), la Marche aléatoire (Backstrom et Leskovec 2011), …

1.3.3.1.3. Apprentissage relationnel

L'apprentissage relationnel présente les données en utilisant un modèle relationnel, ensuite il apprend et infère en se basant sur ce modèle. Spécifiquement, un modèle relationnel est constitué de deux composants : un modèle de données relationnelles et un modèle de structure de dépendance (Getoor et Taskar 2007), (Heckerman, et al. 2001).

1.3.3.2. Approches basées sur la proximité

Les méthodes basées sur la proximité remplacent la probabilité de liaison entre les nœuds en utilisant la proximité entre eux. Elles peuvent généralement être regroupées en méthodes basées sur la proximité nodale et les méthodes basées sur la proximité structurelle.

1.3.3.2.1. Proximité nodale

Ce type de proximité calcule une similarité entre les nœuds en se basant sur leurs caractéristiques. Dans ce qui suit nous allons présenter quelques formules de calcul de similarité entre deux nœuds i et j, avec Y_i et Y_j sont les vecteurs de leurs caractéristiques respectivement :

- Cosine (CS) : la similarité de cosinus (Wang, et al. 2011) est donnée par :

$$CS(Y_i, Y_j) = \frac{\sum_k Y_{ik} Y_{jk}}{\sqrt{\sum_k Y_{ik}^2 \sum_k Y_{jk}^2}} \quad (1.1)$$

- Distance Manhattan (MS) : utilisée pour calculer la dissemblance entre les nœuds (Adalı, Sisenda et Magdon-Ismail 2012) est donné par :

$$MD(Y_i, Y_j) = \sum_k |Y_{ik} - Y_{jk}| \quad (1.2)$$

- Coefficient Jaccard (JC) : coefficient de Jaccard est une fonction de similitude utilisée pour les caractéristiques nominales (Kuo, et al. 2013) est donné par :

$$JC(Y_i, Y_j) = \frac{|Y_i \cap Y_j|}{|Y_i \cup Y_j|} \quad (1.3)$$

- KL-divergence (KL) : fonction de similarité adaptée aux caractéristiques numériques des nœuds (Shen, et al. 2006), est donné par :

$$KL(Y_i, Y_j) = \sum_k h_{ik} \log \frac{h_{ik}}{h_{jk}} + h_{jk} \log \frac{h_{jk}}{h_{ik}} \quad (1.4)$$

Pour deux distributions de probabilités discrètes h_{ik} et h_{jk}, la divergence de Kullback–Leibler de h_{ik} par rapport à h_{jk}.

1.3.3.2.2. Proximité structurelle

- **Proximité structurelle à base de voisinage**

La proximité structurelle entre deux nœuds i et j, peut être mesurée en fonction de leurs voisinages. On donne dans ce qui suit quelques formules de calcul de proximité à base de voisinage :

- Voisin Commun (CN, *Common Neighbor*) : CN entre deux nœuds i et j est calculé comme le nombre de leurs voisins en communs (Liben-Nowell et Kleinberg 2007) , donné par :

$$CN_{ij} = |\Gamma_i \cap \Gamma_j| \quad (1.5)$$

Tels que Γ_i est l'ensemble des nœuds à i et Γ_j est l'ensemble de nœuds adjacents à j.

- Adamic/Adar (AA) : elle est étendue de la mesure CN, AA ne donne pas beaucoup importance aux voisins communs plus connectés du nœud (Adamic et Adar 2003), donné par :

$$AA_{ij} = \sum_{v_z \in \Gamma_i \cap \Gamma_j} \frac{1}{\log|\Gamma_z|} \quad (1.6)$$

Où Γ_z est l'ensemble des nœuds adjacents à v.

- L'attachement préférentiel (PA) : cette méthode recherche la probabilité entre les nœuds fortement corrélés à la taille de leur voisins (Liben-Nowell et Kleinberg 2007), donné par :

$$PA_{ij} = |\Gamma_i| \times |\Gamma_j| \quad (1.7)$$

- Mesure SimRank (SR) : cette mesure est calculée si deux nœuds sont similaires et si leurs voisins sont similaires (Liu et Lü 2010), donné par :

$$SR_{ij} = \frac{\gamma \cdot \sum v_z \in \Gamma_i \ \sum v_{z'} \in \Gamma_j \, SR_{zz'}}{|\Gamma_i| \cdot |\Gamma_j|} \quad (1.8)$$

$SR_{zz'}$ représente la similarité de SimRank entre deux voisins z et z'.

Tels que Γ_i est l'ensemble des nœuds d'utilisateurs adjacents à Γ_j est l'ensemble des nœuds d'items.

- Coefficient Jaccard (voisinage) : il existe une version basée sur le voisinage, appelée, voisinage normalisé (Kuo, et al. 2013), donnée par :

$$JC = \frac{|\Gamma_i \cap \Gamma_j|}{|\Gamma_i \cup \Gamma_j|} \quad (1.9)$$

Tels que Γ_i est l'ensemble des nœuds adjacents à i et Γ_j est l'ensemble de adjacents à j.

1.3.3.2.3. Proximité structurelle à base de chemin

Cette méthode mesure des proximités de chemins cibles reliant les nœuds. Dans ce qui suit nous allons présenter quelques formules de calcul de similarité entre deux nœuds i et j, avec Y_i et Y_j sont les vecteurs de leurs caractéristiques respectivement :

- L'indice Katz (KZ) : calcule la proximité entre les nœuds à l'aide du nombre de chemins les reliant, pondéré par leurs longueurs, est donné par :

$$KZ_{ij} = \sum_{k} \beta^{k} \left| path_{ij}^{\langle k \rangle} \right| \tag{1.10}$$

$path_{ij}^{\langle k \rangle}$ représente le chemin entre deux nœud i et j.

- PageRank (PR) : mesure de proximité structurelle entre $i \; et \; j$ est calculé comme (Zhou, Lu et Zhang 2009) :

$$PR_{ij} = q_i^j + q_j^i \tag{1.11}$$

q_i^j à q_j^i est le PageRank qui relient des nœuds i à j.

- Hitting time : est défini comme le nombre prévu d'étapes nécessaires pour atteindre le nœud j à partir du nœud j pour la première fois, donné par :

$$H_{ij} = 1 + \sum_{k} P_{ik} H_{kj} \tag{1.12}$$

$P_{ik}H_{kj}$: Le nombre prévu de pas qu'il faudrait pour une marche aléatoire pour atteindre i de j.

- Average commute time (ACT) : est une variation symétrique de « Hitting time », donné par (Fouss, et al. 2007) :

$$ACT'_{ij} = H_{ij} \cdot \pi_j + H_{ji} \cdot \pi_i \tag{1.13}$$

$H_{ij} \cdot \pi_j$: nombre moyen d'étapes nécessaires pour visiter nœud i à nœud j et

$H_{ji} \cdot \pi_i$: revenir de nœud j à nœud i par la marche aléatoire

Nous allons donner dans le tableau 1.1, un récapitulatif des différentes méthodes utilisées pour la prédiction de liens en se basant sur la classification donnée dans la section 1.3.3. Il existe d'autres méthodes et techniques qui ont abordé le problème de prédiction de liens et qui se sont basées sur l'apprentissage profond (Schlichtkrull, et al. 2018), l'apprentissage par renforcement (Yo, et al. 2018), les modèles flous (Bhawsar et Thakur 2016), …

Approche	Méthodes	Travaux
Approche basée sur l'apprentissage	Méthode basée sur la classification	➢ (GONG, TALWALKAR et MACKEY 2014) ➢ (Li, Fang et Sheng 2017) ➢ (Schlichtkrull, et al. 2018) ➢ (Farashah, et al. 2021)
	Méthode basée sur un modèle probabiliste	➢ (Fu, Chang et Lee 2014) ➢ (Lakshmi et Bhavani 2021)
	Méthode basée sur l'apprentissage relationnel	➢ (Slokom et Ayachi 2018)
Approche basée sur La proximité	Méthode basée sur Proximité nodale	➢ (Li, et al. 2014) ➢ (Chuan, et al. 2017) ➢ (Ai, et al. 2019)
	Méthode basée sur Proximité structurelle (Voisinage)	➢ (Huang, et al. 2013) ➢ (Kart, et al. 2020)
	Méthode basée sur Proximité structurelle (chemin)	➢ (Kaya 2019) ➢ (Kart, et al. 2020)

Tableau 1.1 : *classification des approches de prédiction de liens.*

Dans le tableau 1.2, nous allons présenter quelques avantages et inconvénients des méthodes de proximité.

Mesure de proximité	Avantage	Inconvénient
Cosine (CS)	Même si les deux nœuds similaires sont éloignés par la distance euclidienne, ils pourraient toujours avoir un angle plus petit entre eux. Plus l'angle est petit, plus la similitude est élevée.	Ne prend pas en compte la différence d'échelle de poids des arêtes entre les différents nœuds.
Distance Manhattan (MS)	Il n'y a pas de mouvement diagonal impliqué dans le calcul de la distance.	Donne une valeur de distance plus élevée que la distance euclidienne car elle ne prend pas le chemin le plus court possible.
Coefficient Jaccard (JC)	Souvent utilisé pour comparer la similarité, la dissemblance et la distance entre les nœuds.	L'indice Jaccard est fortement influencé par le nombre de nœuds.
KL-divergence (KL)	La facilité de résolution des problèmes d'optimisation robustes sur le plan de la distribution.	L'utilisation de la divergence KL pour modéliser les ensembles d'ambiguïtés peut ne pas être une directive pratique pour déterminer la taille de l'ensemble d'ambiguïtés.
Voisin Commun (CN)	Vérifier une corrélation entre le nombre de voisins communs.	Prédiction lente car il faut revoir à tous les nœuds à chaque fois.
Adamic/Adar (AA)	Il a été initialement conçu pour mesurer la relation entre les pages d'accueil personnelles.	Si un nœud a beaucoup de voisins, le score auquel il sera attribué est très faible.
L'attachement préférentiel (PA)	-Modèle génératif dynamique. -Permet d'expliquer la loi de puissance des degrés.	-Le choix de paramètres impact sur le graphe obtenu. -Ne permet pas l'ajustement des données.
Mesure SimRank (SR)	Scalabilité à l'utilisation dans différents types de graphes.	Limité au type des données.
L'indice Katz (KZ)	Cette méthode prend en compte les longueurs de tous les chemins entre chaque paire de nœuds.	Katz est une méthode de prédiction basée sur la topologie du graphe entier et donc son calcul est plus complexe que les autres méthodes.
PageRank (PR)	Très populaire à l'utilisation dans les moteurs de recherches comme Google.	N'admet pas d'interprétation probabiliste directe.
Average commute time (ACT)	-Opposé à la distance de chemin la plus courte, il prend en compte tous les chemins entre deux nœuds.	N'est pas symétrique.

Tableau 1.2 : *Comparaison entre les différentes mesures de similarité.*

1.4. Conclusion

Dans ce chapitre, nous avons vu les notions de base de la théorie des graphes, nous avons aussi étudié la prédiction de liens ainsi que leurs domaines d'application, et on a terminé ce chapitre par un tableau récapitulatif de toutes les méthodes utilisées dans la prédiction de liens.

Nous allons présenter dans le chapitre suivant un état de l'art sur les systèmes de recommandation.

Chapitre 2 : Les systèmes de recommandation

2.1. Introduction

Ce chapitre se divise en deux parties, dans la première partie nous allons présenter des généralités sur les systèmes de recommandation, leur types et méthodes d'évaluation. La deuxième partie concerne l'utilisation de prédiction de liens pour les systèmes de recommandation, où nous allons donner une synthèse des travaux existants sur l'utilisation de prédiction de liens pour les systèmes de recommandations.

2.2. Les systèmes de recommandation : fondement théorique

2.2.1. Définition et concepts de base

Il existe dans la littérature plusieurs définitions des systèmes de recommandation en raison de la diversité des catégorisations de ces systèmes, mais il existe une définition générale de Robin Burke (Burke 2002) qui les définit comme suit :

"*Des systèmes capables de fournir des recommandations personnalisées permettant de guider l'utilisateur vers des ressources intéressantes et utiles au sein d'un espace de données important*".

Un système de recommandation a pour objectif de fournir à un utilisateur des ressources pertinentes en fonction de ses préférences. Ce dernier va ainsi réduire son temps de recherche mais il reçoit également des suggestions de la part du système auxquelles il n'aurait pas spontanément prêté attention (Béchet 2012). Les SR identifient automatiquement les préférences des utilisateurs à travers leurs interactions avec le système, pour leur suggérer des recommandations en utilisant le filtrage d'information. Les deux concepts de base qui apparaissent dans tous les systèmes de recommandations sont l'utilisateur et l'item.

- *Utilisateur* : est la personne qui interagit avec le système, pour laquelle le système va recommander des items en se basant sur ses préférences et ses traces. On note l'ensemble des utilisateurs par U, où un utilisateur donné $u \in U$.
- *Item* : dans les systèmes de recommandation, un item est l'entité qui représente tout élément constituant une liste de recommandation et qui correspond aux besoins de l'utilisateur, incluant tout produit susceptible d'être vendu, emprunté, vu, lu ou écouté. On note l'ensemble des items disponibles dans le système par I, où $i \in I$.
- 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, appelée *note, vote, préférence* ou encore *une évaluation.* Elle consiste en une valeur numérique dans une échelle quelconque (la plus utilisée est [1-5]) ou binaire (aimer \ Ne pas aimer, bon \ mauvais, etc.) qui représente la préférence ou non d'un item donné par un utilisateur. Ces évaluations sont, en général, stockées dans une matrice, qu'on note *R* (pour *Rating*) où : $R : U \times I$.

  On note l'évaluation donnée par un utilisateur u à un item i par $r_{u,i}$ (le *rating* donné par l'utilisateur u à l'item i) qui interprète le triplet $< u, i, r >$, où, une note de 5, par exemple, exprime une grande préférence et une note de 1 indique une faible préférence i.e. l'utilisateur n'a pas aimé l'item. Une note peut être attribuée directement par un utilisateur à un item en donnant une valeur numérique ou binaire à travers l'interface du système appelée *évaluation explicite*.

2.2.2. Les techniques de recommandation

Afin de générer des recommandations, un certain nombre d'approches de filtrage ont été présentées dans la littérature et qui ont été classée dans différentes catégories.

Lakshmi et Bhavani (Lakshmi et Bhavani, Link Prediction Approach to Recommender Systems 2021) proposent de classer les techniques de recommandation en deux grandes classes :

- La recommandation orientée vers une perspective du domaine.
- La recommandation orientée vers une perspective algorithmique qui explore plusieurs volets théoriques.

La figure 2.1 présente la catégorisation proposée par (Lakshmi et Bhavani 2021).

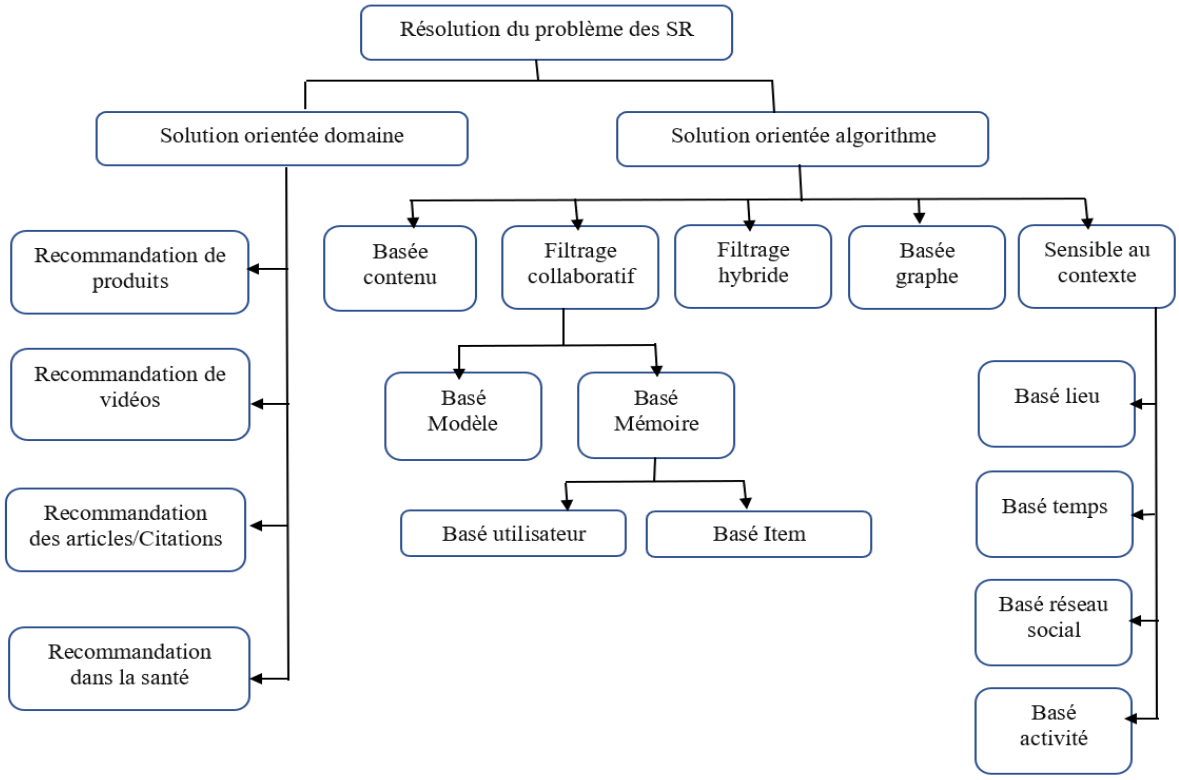


Figure 2.1 : *Catégorisation des systèmes de recommandation d'après* (Lakshmi et Bhavani 2021).

S

La catégorisation la plus classique et la plus répandue est celle qui classe les SR en : Filtrage Collaboratif (FC), Filtrage basé sur le Contenu (FBC) et le filtrage hybride qui consiste en une hybridation des forces et des faiblesses du FC et FBC (Burke 2002).

Dans ce qui suit, nous allons détailler ces trois classes.

2.2.2.1. Filtrage basé sur le contenu

Cette approche se fonde sur une comparaison entre le contenu des items ayant des caractéristiques définies et le profil de l'utilisateur. Chaque utilisateur du système possède un profil qui est généralement basée sur les éléments qu'il a aimés précédemment ou exprimé sous forme d'une liste d'intérêts.

Un système de recommandation basé sur le contenu essaie de recommander à l'utilisateur courant des items (documents, produits, pages web, article de journal, film, musique, etc.) ayant une description similaire au profil d'utilisateur.

Deux fonctionnalités centrales ressortent, pour un système de filtrage basé sur le contenu (Maatallah 2016) :

- La sélection des items : le système fait correspondre les caractéristiques des items avec le profil de l'utilisateur pour décider de sa pertinence ou non.
- La mise à jour du profil : en fonction du retour de pertinence fourni par l'utilisateur sur les items qu'il a reçus, la mise à jour se fait par intégration des thèmes abordés dans les items jugés pertinents.

2.2.2.2. 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. Elles calculent généralement la similarité ou la corrélation globale entre les utilisateurs, et utilisent cela comme poids lors de la formulation de recommandations. 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 : à base de mémoire (appelée aussi à base de : heuristique, voisinage proche) et à base de modèles.

2.2.2.2.1. Collaboration à base de mémoire

Dans la collaboration à base de mémoire, les notes des utilisateurs pour les items qui sont stockées 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. Le calcul de la similarité se fait soit sur l'ensemble des utilisateurs soit sur

l'ensemble des items. On explique ci-dessous le calcul de la similarité et de la prédiction pour les deux types de filtrage.

La figure 2.2 résume la collaboration à base mémoire.

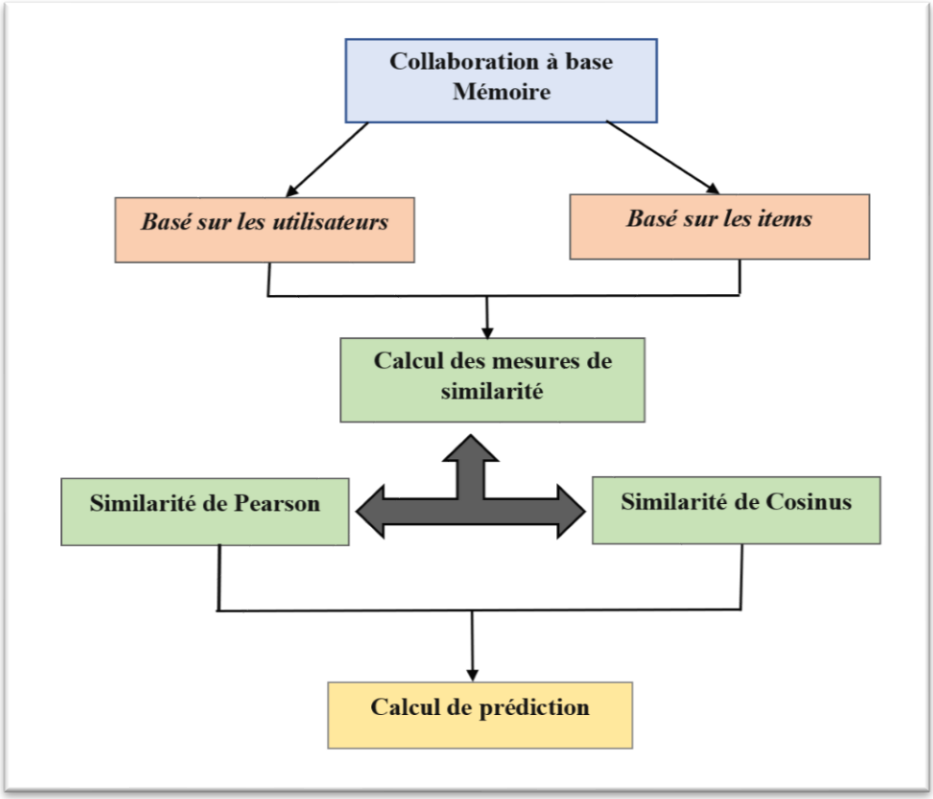


Figure 2.2 : *Filtrage collaboratif basé mémoire.*

- ***Filtrage basé sur les utilisateurs***

Il s'agit de 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.

La similarité entre deux utilisateurs $u \; et \; u'$ peut être calculée en utilisant plusieurs mesures de similarités mais les plus utilisés dans la communauté scientifique des SR sont la similarité de *cosinus* et la corrélation de *pearson*.

- *La similarité de cosinus*

$$sim(u,u') = \cos(v_u, v_{u'}) = \frac{\sum_{i \in I_{uu'}} r_{u,i} \times r_{u',i}}{\sqrt{\sum_{i \in I_{uu'}} r_{u,i}^2} \cdot \sqrt{\sum_{i \in I_{uu'}} r_{u',i}^2}} \quad (2.1)$$

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. $I_{uu'}$ consiste en l'ensemble des items notés à la fois par l'utilisateur u et l'utilisateur u'.

- *La corrélation de Pearson*

$$sim(u,u') = Pearson(u,u') \frac{\sum_{i \in I_{uu'}} (r_{u,i} - \overline{r_u}) \cdot (r_{u',i} - \overline{r_{u'}})}{\sqrt{\sum_{i \in I_{uu'}} (r_{u,i} - \overline{r_u})^2} \cdot \sqrt{\sum_{i \in I_{uu'}} (r_{u',i} - \overline{r_{u'}})^2}} \quad (2.2)$$

Où $\overline{r_u} \; et \; \overline{r_{u'}}$ représentent les moyennes des notes données par les utilisateurs $u \; et \; u'$ respectivement sur les items qu'ils ont notés.

- *Calcul de la prédiction*

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

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

$$p = \overline{r_u} + \frac{\sum_{k \in N(u) \cap U_i} sim(k,u) \times (r_{k,i} - \overline{r_k})}{\sum_{k \in N(u) \cap U_i} |sim(k,u)|} \quad (2.3)$$

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

- ***Filtrage basé sur les items***

Cette méthode et l'inverse de filtrage basé sur l'utilisateur. C'est-à-dire on calcule la similarité entre les items.

- *La similarité de cosinus* : la similarité entre deux items *i* et j est donnée par :

$$sim(i,j) = \cos(v_i, v_j) = \frac{\sum_{u \in U_{ij}} r_{u,i} \times r_{u,j}}{\sqrt{\sum_{u \in U_{ij}} r_{u,i}^2} \cdot \sqrt{\sum_{u \in U_{ij}} r_{u \;\; ,j}^2}} \quad (2.4)$$

Où U_{ij} est l'ensemble des utilisateurs ayant notés les items i et j.

- *Calcul de la prédiction :* la prédiction de note qu'un utilisateur u peut donner à un item i est donnée par :

$$p(u,i) = \frac{\sum_{i \in I_{u(i)}} sim(i,j) \times r_{uj}}{\sum_{j \in I_{u(i)}} |sim(i,j)|} \quad (2.5)$$

2.2.2.2.2. Collaboration à base modèle

Ce modèle est à l'opposé des systèmes basés sur la mémoire. Les systèmes basés sur des modèles utilisent les notes pour apprendre un modèle de prévision. Par la suite, ils utilisent les données disponibles pour entraîner le modèle, qui est ensuite utilisé pour prédire les évaluations des utilisateurs pour les nouveaux items. L'avantage est que ces modèles peuvent être construits hors ligne et peuvent être rapidement utilisés pour calculer des recommandations en ligne.

Les algorithmes de cette approche peuvent être décomposés en trois sous-types : clustering, factorisation matricielle (MF) et l'apprentissage en profondeur (en Anglais Deep Learning).

Chacune de ces approches contient des algorithmes de différents types présentés dans la figure 2.2 ci-dessous.

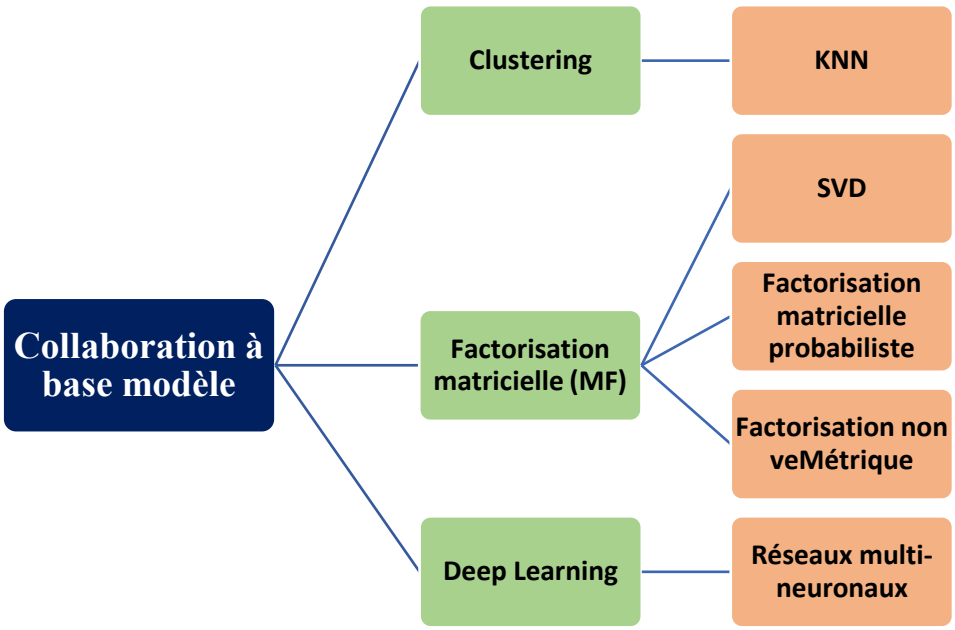


Figure 2.3 : *Types d'approches de filtrage collaboratif basé sur des modèles [1].*

- **Clustering (KNN) :** l'idée est la même que celle des systèmes de recommandation basés sur la mémoire. Dans les algorithmes basés sur la mémoire, nous utilisons les similarités entre les utilisateurs et / ou les items et nous les utilisons comme pondérations pour prédire une note pour un utilisateur et un item. La différence est que les similitudes dans cette approche sont calculées sur la base d'un modèle d'apprentissage non supervisé, plutôt que sur la corrélation de Pearson ou la similarité cosinus. Dans cette approche, il est impératif de limiter le nombre d'utilisateurs similaires à *k*, ce qui rend le système plus évolutif [1].
- **Factorisation matricielle (MF) :** l'idée derrière de tels modèles est que les attitudes ou les préférences d'un utilisateur peuvent être déterminées par un petit nombre de facteurs cachés appelés incorporations (*Embedding*) [1].
- **Deep Learning :** dans un réseau neuronal, on veut que le modèle apprenne les valeurs de la matrice de notes lui-même. Les caractéristiques latentes de l'utilisateur et les caractéristiques latentes de l'item sont recherchées à partir des matrices de notes pour une combinaison spécifique d'item-utilisateur. Elles vont présenter les valeurs d'entrée pour les autres couches linéaires et non linéaires. Cette entrée est passée à plusieurs couches, linéaires ou sigmoïdes et le réseau va apprendre les poids correspondants par n'importe quel algorithme d'optimisation (WANG, LE et GONG 2020).

2.2.2.3. Filtrage hybride

Un système de recommandation hybride est un système qui combine la recommandation basée sur contenu et la recommandation collaborative. L'hybridation de ces deux techniques, afin de traiter les insuffisances de chaque technique et profiter de leurs points forts, a fait l'objet de plusieurs travaux de recherche. Par exemple, il peut s'agir de déterminer les items les plus proches des items appréciés par l'utilisateur en appliquant un filtrage sur le contenu, puis d'appliquer un filtrage collaboratif en se basant sur la qualité des items à partir des évaluations des utilisateurs.

Il existe plusieurs façons de faire de l'hybridation et aucun consensus n'a été défini par la communauté des chercheurs. Toutefois, Burke (Burke 2007) a identifié sept différentes manières de faire de l'hybridation :

- Pondéré : le score ou le vote obtenu par chacune des deux techniques est combiné en un seul résultat.

- Sélection : le système bascule entre les deux techniques de recommandation en fonction de la situation.
- Mixte : les recommandations des deux techniques sont proposées simultanément.
- Combiner les propriétés : les données issues des deux techniques sont combinées et transmises à un seul algorithme de recommandation.
- Augmentation de propriétés : le résultat d'une technique est utilisé comme entrée de l'autre technique.
- Cascade : un système affine les recommandations données par l'autre système.
- Meta-level : une première technique construit un modèle qui sera utilisé comme entrée par la seconde technique.

2.2.3. Evaluation des systèmes de recommandation

Selon (Herlocker, et al. 2004) l'évaluation de performance des systèmes de recommandation est souvent limitée à la précision de prédiction et la qualité de recommandation et peut être effectuée en utilisant une analyse hors ligne, une expérimentation avec des utilisateurs réels, ou en ligne.

❖ **Évaluation hors ligne**

L'évaluation hors ligne est l'évaluation la plus simple à réaliser et la moins risquée. Fondamentalement, il s'agit de diviser les données disponibles en une partie d'apprentissage et une partie de test et utiliser la partie d'apprentissage pour prédire la partie de test. Elle facilite l'intégration d'un grand nombre d'utilisateurs mais elle n'est pas très sensible aux changements potentiels du comportement des utilisateurs (Shani et Gunawardana 2011).

L'évaluation se fait en calculant certaines mesures et erreurs à l'image du : MAE, NMAE, RMSE et d'autre. Dans ce qui suit, nous citerons quelques mesures de l'évaluation.

- La mesure de l'erreur absolue moyenne (MAE : Mean Absolute Error) c'est une mesure statistique qui s'appuie sur la moyenne des différences entre chaque note prédite et sa note réelle, formellement :

$$MAE = \frac{\sum_{(u,i)\in K}|r_{u,i} - \hat{r}_{u,i}|}{|K|} \quad (2.6)$$

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

$\hat{r}_{u,i}$est la note prédite par le SR.

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

- La mesure NMAE (Normalized mean absolute error) : normalise les erreurs en divisant par l'ensemble de tous les couples (utilisateurn item) possibles (r_{high} et r_{low} sont respectivement les possibilités maximale et minimale du système), ce qui donne une métrique dans l'intervalle [0,1] pour toutes les échelles de notation, formellement :

$$NMAE = \frac{1}{K(r_{high} - r_{low})} \sum_{u,i}|r_{u,i} - \hat{r}_{u,i}| \quad (2.7)$$

- RMSE (Root mean squared error) est une mesure connexe qui a pour effet de mettre davantage l'accent sur les grosses erreurs. Elle est calculée comme MAE, mais elle corrige l'erreur avant de l'additionner, sa formule :

$$RMSE = \sqrt{\frac{1}{K}\sum_{u,i}\left(r_{u,i} - \hat{r}_{u,i}\right)^2} \quad (2.8)$$

❖ **Etudes sur un échantillon d'utilisateurs**

Cette méthode consiste à recruter un groupe de volontaires qui doivent utiliser le SR pour effectuer des tâches spécifiques, il faut surveiller et enregistrer leur comportement pendant l'expérience. Ensuite, vous pouvez poser des questions aux participants sur leurs impressions sur l'expérience. Ce type de test permet de suivre le comportement de l'utilisateur lors de l'interaction avec le SR, et d'observer si ce dernier affecte le comportement de l'utilisateur (Silveira, et al. 2019).

❖ **Evaluation en ligne**

Elle est appliquée aux utilisateurs réels du système en temps réel. Ce test peut simplement comparer le taux des chiffres d'affaires avant et après l'application de SR (Kohavi, et al. 2009).

Cette évaluation comporte des risques. En effet, si le système recommande des éléments non pertinents, nous risquons de perdre des utilisateurs (Shani et Gunawardana 2011).

2.2.4. Limites des systèmes de recommandation

Les systèmes de recommandation se basant sur les techniques précédemment expliquées ont certaines limites (Adomavicius et Tuzhilin 2005). Ci-dessous les problèmes les plus importants sont décrits :

- *Démarrage à froid :* le système de filtrage collaboratif dépend de la note de l'utilisateur de l'item. Par conséquent, aucun nouvel item ne peut être recommandé tant qu'il n'y a pas d'avis (note) d'utilisateurs. Dans un système de recommandation basé sur un filtrage collaboratif et des systèmes basés sur le contenu, si l'historique d'évaluation des items de l'utilisateur n'est pas connue, les préférences de l'utilisateur ne peuvent pas être prédites. Par conséquent, les nouveaux utilisateurs ne recevront pas de suggestions spécifiques tant qu'ils n'auront pas évalué un certain nombre d'items (TADLAOUI 2018).
- *Parcimonie (sparsity) :* lorsque le nombre d'items notés par l'utilisateur est très faible par rapport au nombre total d'items dans le système, le système de recommandation devient dispersé. Ce fait conduit à une très faible densité de la matrice d'évaluation utilisateur / item. Cela a un impact sur la capacité du système de recommandation à recommander tous les items disponibles et sur l'exactitude des recommandations générées (TADLAOUI 2018)
- *Sérendipité :* étant donné que le système de recommandation basé sur le contenu ne recommande que les éléments qui correspondent au profil de l'utilisateur, les utilisateurs ne recevront que des recommandations similaires à celles qu'ils ont déjà rencontrées. Il n'aura aucune chance de recevoir des suggestions inattendues. Cela peut amener les utilisateurs à se lasser des suggestions (TADLAOUI 2018).
- *Problème du mouton gris :* les utilisateurs de systèmes de recommandation peuvent avoir des préférences spéciales et des préférences très inhabituelles, ces utilisateurs sont situés à la limite entre deux ou plusieurs clusters d'utilisateurs. Cela les empêche de trouver des utilisateurs similaires et des suggestions associées (Mohamed, Khafagy et Ibrahim 2019).

- *Montée en charge :* plus il y a d'utilisateurs et des items dans le système de recommandation, plus les calculs requis pour la recommandation sont coûteux. (Mohamed, Khafagy et Ibrahim 2019).

2.3. Prédiction de liens et système de recommandation

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

- Motivation : l'inspiration et la raison pour laquelle l'auteur choisit l'approche de prédiction de liens.
- Domaine d'application : Le type de données utilisées pour valider le travail (base de données, réseau social, etc.).
- Type de graphe : le graphe utilisé dans la recherche s'il est mono-partie, biparti ou plusieurs graphes.
- Approche : l'approche utilisée pour la prédiction de liens (apprentissage ou proximité).
- Méthode : quelle est la méthode utilisé (classification, modèle probabiliste, proximité nodale, ... etc.).
- Algorithme : les algorithmes utilisés.

Référence	Motivation	Domaine d'application	Type du Graphe	Approche	Méthode	Algorithme
(Benchettara, Kanawati et Rouveirol 2010)	-Améliorer les performances du système de recommandation	DBLP	Biparti + Projection	Apprentissage + Proximité	Classification -Proximité nodale - Proximité structurelle (Voisinage + chemin)	CN, JC, AA, PA, Katz, plus court chemin, PR, Commute-time
(Huang, et al. 2013)	-Démarrage à froid -Réutilisation des services	ProgrammableWeb data	Mono-parti	Proximité	Proximité structurelle (Voisinage)	Prédiction de liens basée sur l'agrégation du rang
(Xie, et al. 2014)	-Sparsity	-MovieLens -AppChina	Biparti	Apprentissage	-	Nouvel Algorithme
(Fu, Chang et Lee 2014)	-le calcul d'une mesure de proximité appropriée entre deux nœuds	Digital Bibliography & Library Project (DBLP)	Biparti + Projection	Apprentissage	Modèle probabiliste	Marche aléatoire
(Li, et al. 2014)	-Sparsity -Sérendipité	BDD FoodMart	Biparti	Proximité	Proximité nodale	DB_Coefficient Jaccard
(Garg, Viswanathan et Desai 2015)	-Pour améliorer la prédiction des liens entre les utilisateurs et les items.	-BDD Vidéo d'Amazon -BDD de Téléphones portables et accessoires -BDD de livres	Mono-parti + Biparti	Proximité de communautés	Proximité nodale	-Infomap -PR -Centralité des valeurs propres

(Chuan, et al. 2017)	-Améliorer la recommandation	-hep-th archive -hep-lat archive -AMC	Mono-parti	Proximité	Proximité nodale	Nouvel Algorithme
(Talasu, et al. 2017)	- Sérendipité -Démarrage à froid	ID3 dans le domaine de musique	Biparti	Proximité de communautés	Proximité nodale	Prédiction de liens basée détection de communautés.
(Slokom et Ayachi 2018)	-résoudre la surcharge d'informations dans les réseaux sociaux et créer des opportunités.	Réseau social	Biparti	Apprentissage	Modèle probabiliste relationnel	-Modèle de données relationnelles -Marche aléatoire
(Su, Zheng, et al. 2019)	- Degré de confiance dans le résultat de prédiction.	MovieLens	Biparti	Proximité	-	Nouvel Algorithme
(Kaya 2019)	-Démarrage à froid -Sparsity	BDD de TripAdvisor.com	Biparti	Apprentissage + Proximité	Classification -Proximité nodale - Proximité structurelle (Voisinage + chemin)	CN, JC, PA, Allocation des ressources
(Ai, et al. 2019)	-Améliorer la prédictions -Démarrage à froid	MovieLens	Mono-parti + Pondéré	Proximité de communautés	Proximité nodale	Nouvel Algorithme

(Kart, et al. 2020)	-capturer plus précisément le monde réel et limiter les lacunes des données de transaction conventionnelles.	-Goodreads -MovieLens	Biparti + Pondéré	Apprentissage + Proximité	Classification, Proximité nodale, Proximité structurelle (Voisinage + chemin)	JC, AA, PA, Chemin local
(Su, Zheng, et al. 2020)	-Obtenir une meilleure précision prédictive.	-MovieLens -Netflix	Biparti	Apprentissage + Proximité	-	Nouvel Algorithme
(Kou, et al. 2020)	-Sparsity -protection des informations de confidentialité	Epinions	Mono-parti	Apprentissage + Proximité	Proximité nodale	-Simhash technology -Fuzzy computing
(Lakshmi et Bhavani 2021)	-Sparsity	MovieLens	Biparti	Apprentissage	Modèle probabiliste	Bipartite Co-occurrence Probability (B_COP)
(Farashah, et al. 2021)	-Démarrage à froid	MovieLens	Biparti + Projection	Apprentissage + Proximité	Classification	-DBScan clustering -Deep neural network Friendlink

Tableau 2.1 : *Synthèse des travaux sur la recommandation et la prédiction de liens.*

2.4. Conclusion

Dans ce chapitre, nous avons étudié les systèmes de recommandation ainsi que leurs types, les mesures d'évaluation pour un système de recommandation, et leurs problèmes, et on a terminé ce chapitre par quelques travaux sur la prédiction de liens appliquée dans les systèmes de recommandation.

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

Chapitre 3 : Un système de recommandation basé prédiction de liens

3.1. Introduction

Dans ce chapitre, nous allons décrire l'approche que nous proposons pour la recommandation. Nous commençons d'abord par une présentation de notre problématique et l'objectif de notre contribution. Par la suite, nous allons présenter l'architecture générale de notre système ainsi qu'une description détaillée de chacune de ses phases : de la collecte de données jusqu'au calcul de la recommandation.

3.2. Problématique et objectifs de notre approche

Les systèmes de recommandation (SR) sont un outil de recherche et de filtrage d'information qui visent à proposer aux utilisateurs des items qui pourraient les intéresser.

La technique des SR à laquelle nous nous intéressons dans notre travail est le filtrage collaboratif (FC). Elle est considérée comme l'une des techniques les plus utilisées dans les systèmes de recommandation, en raison de son efficacité ((Bobadilla, et al. 2013) ; (Silveira, et al. 2019)). Cette technique repose sur une hypothèse fondamentale stipulant que les utilisateurs ayant noté, aimé les mêmes items ou ayant des comportements similaires (acheter, regarder, consulter, …), dans le passé, ils évalueront ou agiront sur d'autres items de la même manière, dans le futur. En effet, cette technique se base sur une matrice d'évaluations (notes) exprimées par les utilisateurs sur les items qui les ont intéressés.

Cependant, en pratique, un grand nombre de notes de cette matrice n'est pas disponible, en raison de la rareté inhérente aux données d'évaluation, ce qui est plus connu sous le nom du problème de parcimonie (*sparsity*, en anglais). Par conséquent, la qualité de la prédiction de notes baisse d'une manière considérable ce qui réduit la précision des algorithmes de FC.

En outre, les algorithmes de filtrage collaboratif tel qu'ils sont ne prennent pas en considération l'arrivée d'un nouvel utilisateur ou un nouvel item qui n'a pas d'historique de préférences ou de

notes dans la matrice d'évaluations. Ceci est plus connu sous le nom du problème de démarrage à froid.

Ces deux problèmes limitent considérablement l'applicabilité des systèmes de recommandation en particulier dans les applications à grande échelle.

Pour surmonter ces problèmes, l'axe de la recommandation basée sur les graphes a été exploré afin de bénéficier des relations cachées et des corrélations qui existent entre les différents types de nœuds. Ces corrélations ne sont principalement pas basées sur la notion d'évaluation.

L'objectif principal de notre étude consiste à l'exploration des approches topologiques et des structures graphiques pour le calcul de recommandation afin de pallier au problème de rareté des données ainsi qu'au problème d'intégration d'un nouvel item.

Afin d'atteindre notre objectif, nous proposons une approche divisée en deux grandes parties.

La première va avoir comme sous-objectif la résolution du problème d'arrivée d'un nouvel item sans historique d'évaluation. Pour ce faire, nous proposons d'explorer les relations existantes entre les items afin de construire un graphe de connaissance avec un seul type de nœuds (items) et une relation entre ces nœuds différentes de la relation d'évaluation. A partir de ce graphe, nous allons collecter des informations liées aux corrélations latentes entre les nœuds, ceci sera réaliser en utilisant les techniques de détection de communautés (Fortunato 2010) afin de classer les items qui sont fortement liés entre eux dans des communautés. Ces informations seront injectées à la deuxième partie afin d'aider au processus de recommandation.

La deuxième partie se focalise sur la recommandation basée graphe. En effet, la matrice d'évaluation peut être vue comme une matrice d'adjacence d'un graphe biparti avec comme types de nœuds : les utilisateurs et les items. Le problème de recommandation se traduit alors en un problème de prédiction de liens (Liben-Nowell et Kleinberg 2007) dans ce graphe biparti.

En effet, la prédiction de liens fournit des méthodes pour estimer les connexions potentielles dans les graphes, ce qui a une importance théorique et pratique pour la recommandation. Notre objectif, à ce stade, sera d'adapter les techniques de prédiction de liens afin de prendre en compte les spécificités du graphe biparti.

3.3. Modèle proposé

Le modèle de recommandation que nous proposons est basé sur l'utilisation des graphes. Ces derniers ont récemment prouvé leur efficacité, dans ce cadre, dans le sens où ils peuvent nous apprendre davantage d'informations latentes sur les nœuds et les relations entre eux.

Un graphe biparti sera construit en se basant sur la matrice d'évaluation $R: U \times I$ avec deux types de nœuds : utilisateurs et items, ce qui nous amène à explorer les méthodes de prédiction de liens pour améliorer la recommandation.

Cependant, ce graphe biparti construit à partir de R souffre du même problème qui se pose avec cette matrice à savoir l'intégration d'un nouvel item (utilisateur) qui ne possède pas d'historique de notes.

Pour solutionner ce problème, nous proposons d'explorer les corrélations existantes entre les nœuds du même type. L'idée est que ces nœuds peuvent avoir des relations de différents degrés : social, démographique, ou encore lié au domaine d'étude. Nous allons bénéficier de ces relations afin de construire un deuxième graphe avec le même type de nœud (les items dans notre cas), ce qui va nous aider à classer les items en se basant sur des relations autres que les notes.

L'ensemble de données que nous allons utiliser est tiré de la base de données MoviLens[1]. Elle consiste en une base de données des films largement utilisée dans ce domaine. Plus de détails de la BDD seront donnés dans le chapitre suivant.

L'architecture du système proposé est donnée dans la figure 3.1 constituée de quatre phases que nous allons détailler dans ce qui suit.

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

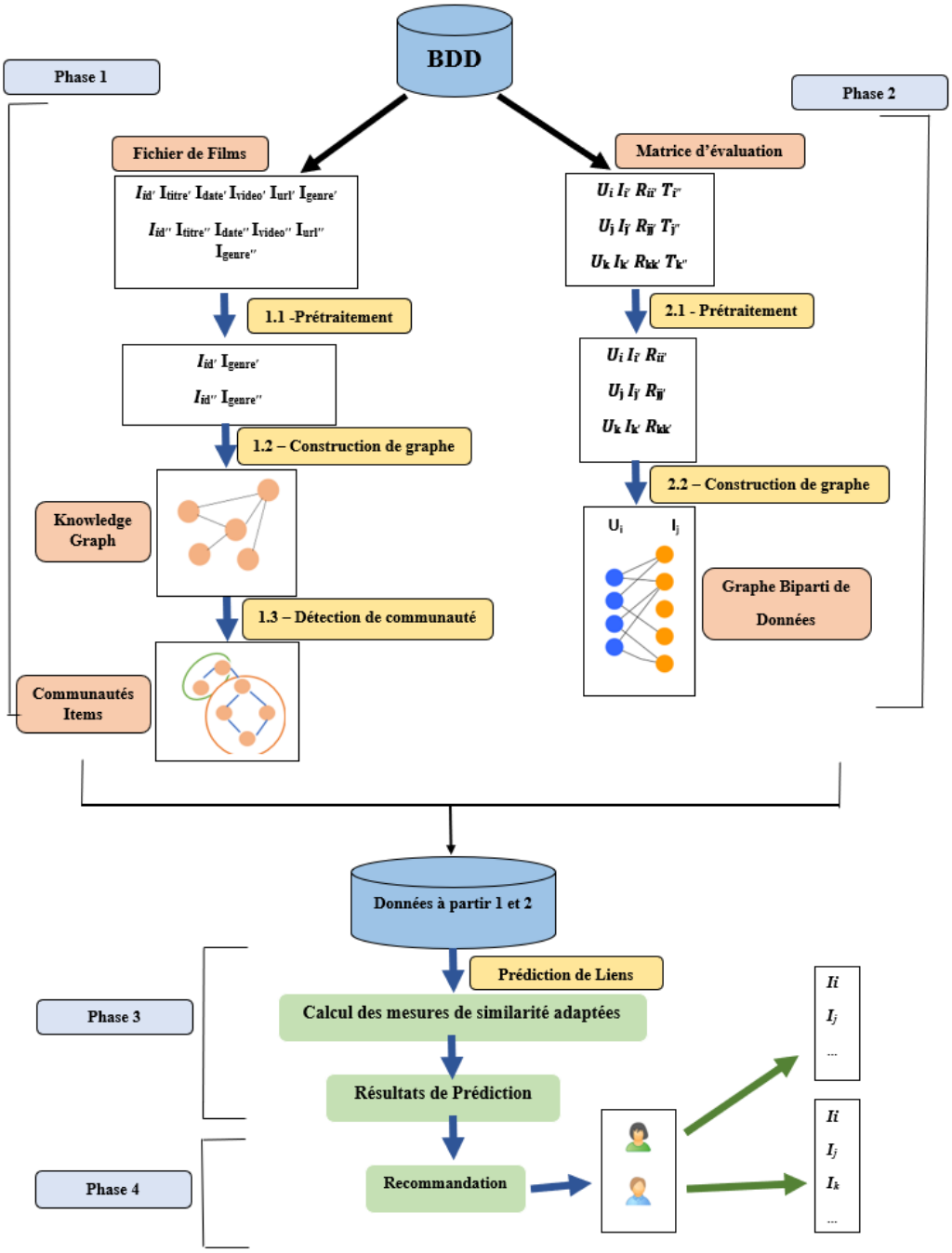


Figure 3.1 : *Architecture du système de recommandation proposé.*

3.3.1. Phase 1 : Exploration et préparation des données

Cette phase a deux principaux objectifs, le premier consiste à pallier au problème de rajout d'un nouvel item et le deuxième consiste à enrichir la phase 3 avec des informations utiles pour le calcul de prédiction de liens.

Lors de cette phase, nous allons, dans un premier temps, construire un graphe de connaissance (knowledge graph) (Wang, et al. 2017) sur l'ensemble des films. Dans un second temps, sur ce graphe de connaissance, nous allons appliquer les techniques de détection de communautés ((Fortunato 2010) ; (Fortunato et Hric 2016)) afin de trouver des communautés de films indépendamment de leurs évaluations.

Cette phase se divise en trois sous-étapes : pré-traitement, construction du graphe de connaissance, détection de communautés.

3.3.1.1. Pré-traitement

La base de données Movielens contient un fichier informatif sur l'ensemble des films. Ce fichier (figure 3.2) contient les informations suivantes :

- *Movie id :* l'identifiant du film,
- *Movie title :* le titre du film,
- *Release date :* la date de sortie du film,
- *Video release date :* la date de sortie de la vidéo,
- *IMDb URL :* le lien du film dans Internet Movie Database,
- *Movie genre :* le genre du film. Movielens classe les films dans 19 genres différents (inconnu, action, aventure, animation, …), un 1 indique que le film est de ce genre, un 0 indique qu'il n'est pas. De plus, un film peut avoir plusieurs genres.

```
1|Toy Story (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?Toy%20Story%20(1995)|0|0|0|1|1|1|0|0|0|0|0|0|0|0|0|0|0
2|GoldenEye (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?GoldenEye%20(1995)|0|1|1|0|0|0|0|0|0|0|0|0|0|0|1|0|0
3|Four Rooms (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?Four%20Rooms%20(1995)|0|0|0|0|0|0|0|0|0|0|0|0|0|0|1|0|0
4|Get Shorty (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?Get%20Shorty%20(1995)|0|1|0|0|0|1|0|0|1|0|0|0|0|0|0|0|0
5|Copycat (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?Copycat%20(1995)|0|0|0|0|0|0|0|1|0|1|0|0|0|0|0|0|0|1|0|0
6|Shanghai Triad (Yao a yao yao dao waipo qiao) (1995)|01-Jan-1995||http://us.imdb.com/Title?Yao+a+yao+yao+dao+waipo+qiao+(1995)|0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0|0
7|Twelve Monkeys (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?Twelve%20Monkeys%20(1995)|0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|1|0|0|0
8|Babe (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?Babe%20(1995)|0|0|0|0|0|1|1|0|0|1|0|0|0|0|0|0|0|0|0
9|Dead Man Walking (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?Dead%20Man%20Walking%20(1995)|0|0|0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|0|0
10|Richard III (1995)|22-Jan-1996||http://us.imdb.com/M/title-exact?Richard%20III%20(1995)|0|0|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|1|0
11|Seven (Se7en) (1995)|01-Jan-1995||http://us.imdb.com/M/title-exact?Se7en%20(1995)|0|0|0|0|0|0|1|0|0|0|0|0|0|0|0|1|0|0
```

Figure 3.2 : *le fichier informatif de films de Movielens.*

Le graphe de connaissance que nous allons construire contient un seul type de nœuds : **les films** et un seul type de relation : **le genre**. Pour ce faire, nous avons extrait l'identifiant de chaque film ainsi que la liste des genres qui lui sont associés. Ceci résulte en une matrice d'adjacence d'un graphe biparti liant chaque film à son (ses) genre : $I \times Gr$.

Un exemple d'un ensemble de films avec leurs genres est donné dans la figure 3.3.

	g1	g2	g3	g4	g5	g6	g7	g8	g9	g10	g11	g12	g13	g14	g15	g16	g17	g18	g19
i1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
i2	1	0	1	0	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0
i3	0	0	0	0	1	1	0	0	0	1	0	1	0	0	0	0	0	0	0
i4	0	0	0	1	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0

Figure 3.3 : *Un exemple de matrice d'adjacence du graphe film x genres.*

3.3.1.2. Construction du graphe de connaissance

A partir de l'ensemble des films ainsi que de leurs genres, nous allons générer un graphe de connaissance mono-parti avec comme relation entre les nœuds le genre des films selon la règle suivante :

Une arête est générée entre deux items s'ils ont au moins un genre en commun

Exemple : En se basant sur l'exemple de la matrice $I \times Gr$ donnée dans la figure 3.3, on obtient le graphe suivant (figure 3.4) :

	g1	g2	g3	g4	g5	g6	g7	g8	g9	g10	g11	g12	g13	g14	g15	g16	g17	g18	g19
i1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
i2	1	0	1	0	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0
i3	0	0	0	0	1	1	0	0	0	1	0	1	0	0	0	0	0	0	0
i4	0	0	0	1	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0

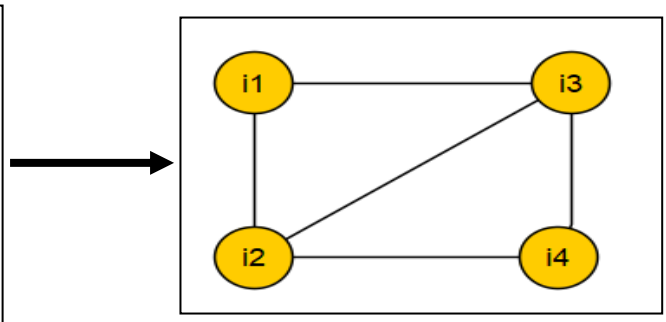


Figure 3.4 : *Graphe de connaissance généré à partir de la matrice* $I \times Gr$.

Dans ce qui suit, nous allons donner le pseudocode de l'algorithme de construction du graphe de connaissance.

Entrées : matrice d'adjacence $I \times Gr$
L_k- lignes de la matrice
Sortie : matrice d'adjacence $I \times I$

Début

$k = 0$

1. **Pour tout** ($L_k \in Lignes\_matrice$) **faire**
2. **Si** ($g_{Lk} = g_{Lk+1}$) et ($g_{Lk} = 1$) et ($g_{Lk+1} = 1$) **alors**
3. Générer une liaison entre i_{Lk} et i_{Lk+1}
4. $k = k + 1$
5. **Fin Si**
6. **Fin Pour**

Fin

3.3.1.3. Détection de communautés

Le graphe construit dans l'étape suivante sera soumis à cette étape afin de générer des classes des items.

Nous avons choisi d'utiliser les techniques de détection de communautés ((Fortunato 2010) ; (Fortunato et Hric 2016)) qui consiste en un processus de découverte des groupes cohésifs de nœuds dans un graphe, appelés communauté. L'ensemble de nœuds appartenant à une communauté sont fortement connectés à l'intérieur de la communauté qu'à l'extérieur.

Puisque la détection de communautés ne présente pas un objectif dans notre travail, ce n'est qu'une phase de préparation pour la prédiction de liens, nous n'allons pas s'étaler à expliquer son fondement théorique et ses différents algorithmes. Plus d'informations liées à cet axe peuvent être trouvées dans ((Fortunato 2010); (Fortunato et Hric 2016) ; (El-Moussaoui, et al. 2019)). Nous allons tester notre approche avec trois algorithmes qui ont prouvé leur efficacité et rapidité dans la littérature. Le premier algorithme a été proposé par (Blondel, Guillaume et Lambiotte 2008), baptisé Louvain, est un algorithme agglomératif basé sur la maximisation de la modularité. Le deuxième algorithme est aussi basé sur la maximisation de modularité et les vecteurs propres, il s'agit de l'algorithme de Newman (Newman 2006). Le troisième algorithme est Infomap (Rosvall, Axelsson et Bergstrom 2009). Ce dernier a un principe différent, il réduit le problème de détection de communautés à un problème de codage où il exploite la compression des données, en compressant le mouvement d'un marcheur aléatoire sur un graphe.

3.3.2. Phase 2 : Génération du graphe biparti

L'objectif de cette phase est de construire un graphe biparti à partir de la matrice d'évaluation $R: U \times I$ qui présente sa matrice d'adjacence. Cette phase est constituée de deux sous étapes.

3.3.2.1. Pré-traitement

Nous allons commencer par effectuer des pré-traitements sur la matrice d'évaluation de la BDD considérée. En effet, cette matrice contient les utilisateurs, les items, les notes que les utilisateurs ont donné aux items ainsi que le temps d'évaluation.

Puisque, dans notre travail, nous ne considérons pas l'aspect temporel du graphe, nous allons filtrer cet ensemble de données afin d'extraire les informations qui nous seront utiles dans la suite du travail.

Pour ce faire, nous avons extrait l'identifiant de chaque utilisateur, l'identifiant de chaque item ainsi que les notes données aux items par ces utilisateurs. Ceci résulte en une matrice d'adjacence d'un graphe biparti liant chaque utilisateur à la liste des items qu'il a noté.

3.3.2.2. Construction du graphe

A partir de la matrice $R: U \times I$ résultante à l'étape de pré-traitement, nous allons générer un graphe biparti reliant chaque utilisateur à l'ensemble des items qu'il a noté selon la règle suivante :

Une arête est générée entre un nœud utilisateur et un nœud item si cet utilisateur a évalué cet item à travers une note.

Exemple : On donne un exemple une matrice $R: U \times I$ qui contient quatre utilisateurs et six items, ainsi que le graphe biparti qui lui correspond dans la figure 3.5.

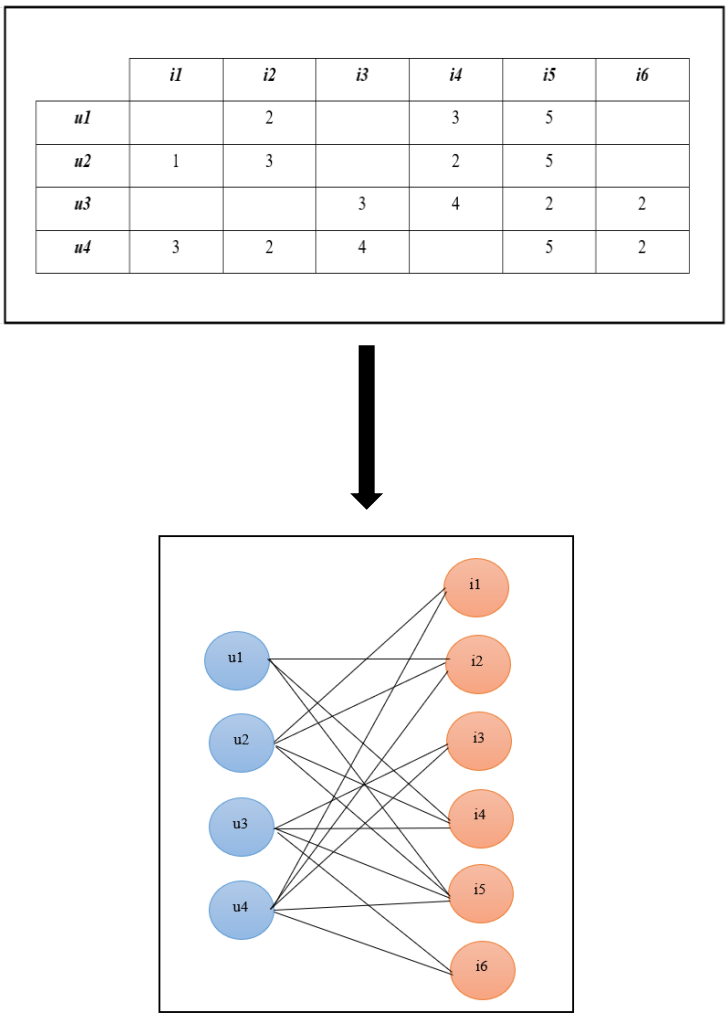


Figure 3.5 : *Un exemple de construction d'un graphe biparti à partir d'une matrice d'évaluation.*

3.3.3. Phase 3 : Prédiction de liens

Cette phase est l'équivalent de la prédiction de notes dans le filtrage collaboratif. Cependant, au lieu de se baser sur la matrice d'évaluation qui est connue d'être clairsemée (sparse), nous allons exploiter la structure graphique de cette matrice et nous allons faire appel aux techniques d'analyse de graphes. Son objectif est de prédire des liens potentiels dans le graphe biparti construit à partir de $R: U \times I$ entre les utilisateurs et les items. Ces liens mesurent l'intérêt qu'un utilisateur peut porter à un item.

Cette phase est constituée de deux sous-phases. La première sert à déterminer l'ensemble des items candidats à la recommandation (à la prédiction de liens) et la deuxième sert à prédire des relations entre chaque utilisateur et sa liste des items candidats.

3.3.3.1. Déterminer les items candidats

Cette étape va bénéficier des données collectées à partir de la phase 1 et la phase 2. En effet, la phase 1 nous a permis de trouver des communautés des items et la phase 2 nous a permis de construire un graphe biparti qui lie chaque utilisateur à l'ensemble des items qu'il a noté.

Pour déterminer l'ensemble des items candidats d'un utilisateur, nous avons besoin des ensembles suivants, tirés des phases 1 et 2 :

Pour un utilisateur cible u, nous notons :

- I_u, l'ensemble des items qu'il a noté,
- $\overline{I_u}$, l'ensemble des items qu'il n'a pas noté,
- C_u, l'ensemble des communautés auxquelles appartiennent les items notés par u.
- $\overline{C_u}$, l'ensemble des communautés auxquelles appartiennent les items que u n'a pas noté.

A ce stade, nous pouvons déterminer l'ensemble de communautés auxquelles appartiennent les items notés et non notés par u en même temps, que nous notons C', qui est le résultat de l'intersection des ensembles C_u et $\overline{C_u}$:

$$C' = C_u \cap \overline{C_u} \tag{3.1}$$

Les items appartenant à l'ensemble C' et qui n'ont pas été notés par u représentent notre ensemble d'items candidats pour l'utilisateur u, que nous notons I_u^{cand}.

Dans ce qui suit, nous allons illustrer notre processus de détermination des items candidats à travers un exemple en considérant la même matrice d'évaluation donnée dans la figure 3.5.

Exemple illustratif

Nous supposons qu'à la fin de la phase 1, nous avons trouvé trois communautés. La première communauté, C_1, contient les items i_1, i_2, i_3 ; la deuxième communauté, C_2, contient les items i_4, i_5 et la troisième communauté, C_3, contient l'item i_6.

	i1	*i2*	*i3*	*i4*	*i5*	*i6*
u1		2		3	5	
u2	1	3		2	5	
u3			3	4	2	2
u4	3	2	4		5	2

Figure 3.6 : *Les communautés des items.*

- Pour l'utilisateur cible u_1, nous déterminons, dans un premier temps :
  - $I_{u_1} = \{i_2, i_4, i_5\}$, l'ensemble des items qu'il a noté,
  - $\overline{I_{u_1}} = \{i_1, i_3, i_6\}$, l'ensemble des items qu'il n'a pas noté,
  - $C_{u_1} = \{C_1, C_2\}$, l'ensemble des communautés auxquelles appartiennent les items qu'il a noté,
  - $\overline{C_{u_1}} = \{C_1, C_3\}$, l'ensemble des communautés auxquelles appartiennent les items qu'il n'a pas noté.
- Nous allons, maintenant déterminer la liste des items candidats pour u_1 :
  - $C'_{u_1} = C_{u_1} \cap \overline{C_{u_1}} = \{C_1\}$
  - $I_{u_1}^{cand} = \{i_1, i_3\}$

Dans ce qui suit nous donnons le pseudo-code de l'algorithme de détermination des items candidats.

Item Candidats

Entrées : List des items non noté avec communautés (C_non_noté) [u_i i_j c_k]

List des items noté avec communautés (C_noté) [u_a i_b c_e]

*L*n- lignes de la matrice non noté avec communautés
*L*m- lignes de la matrice noté avec communautés
Sorites : List des items candidats

Début

n = 0, m = 0

1. **Pour tout** ($L_n \in Lignes\_nonNoté\_ communautés$) **faire**
2. **Pour tout** ($E_m \in Lignes\_noté\_ communautés$) **faire**
3. **Si** ($u_{Ln} = u_{em}$) et (communauté d'item non noté c_{Ln} = communauté d'item noté c_{em}) **alors**
4. i_{ln} est un item candidats de u_{Ln}
5. m = m + 1
6. n = n + 1
7. **Fin Si**
8. **Fin Pour**
9. **Fin Pour**
10. **Fin**

3.3.3.2. Prédiction de liens

Une fois que nous avons déterminer la liste des items candidats pour chaque utilisateur, nous allons à présent procéder au calcul de prédiction de liens entre chaque utilisateur et les items susceptibles à lui intéresser.

Comme nous l'avons présenté dans le chapitre 1, les méthodes de prédiction de liens se divisent en deux grandes familles : basées apprentissage et basée proximité. Dans notre travail, nous nous intéressons à explorer les méthodes basées sur la proximité et plus précisément la proximité nodale.

En effet, toutes les métriques utilisées pour trouver une proximité entre les nœuds proposées dans la littérature sont applicables dans des graphes homogènes (mono-parti avec un seul type de nœud). Dans le cadre des systèmes de recommandation basés graphe, deux types de nœuds sont considérés : les utilisateurs et les items, et l'objectif est de prédire des liens qui sont plus susceptibles de se former dans le graphe à un temps futur entre les deux différents nœuds.

Plusieurs adaptations de ces métriques ont été proposées dans la littérature ((Koptelov, Zimmermann et Cremilleux 2019) ; (Kunegis, Luca et Albayrak 2010) ; (Lakshmi et Bhavani 2018))), que ce soit basées sur d'autres informations liées à la structure du graphe, ou une projection ou encore dépendantes du domaine.

Dans notre travail, nous proposons une adaptation des deux mesures : voisins communs que nous notons (CN_B) et le coefficient de Jaccard que nous notons (JC_B), qui sera basée sur les informations tirées de notre domaine d'application (SR).

3.3.3.2.1. Voisins communs

Pour chaque nœud utilisateur cible u et chaque nœud item candidat de u, i, tiré de l'étape précédente, nous allons calculer le nombre de leurs voisins communs $CN_B(u,i)$, comme suit :

$$CN_B(u,i) = |\{u' \in U | I_{u'} \cap I_u \neq \emptyset, i \in I_{u'}\}| \quad (3.2)$$

Où :

- U, est l'ensemble de tous les utilisateurs,
- $I_{u'}$, I_u, sont les ensembles des items notés par les utilisateurs u' et u respectivement.

Exemple illustratif (suite)

- Pour l'utilisateur cible u_1, nous allons calculer le nombre de voisins communs avec ses items candidats : i_1 et i_3, nous obtenons :

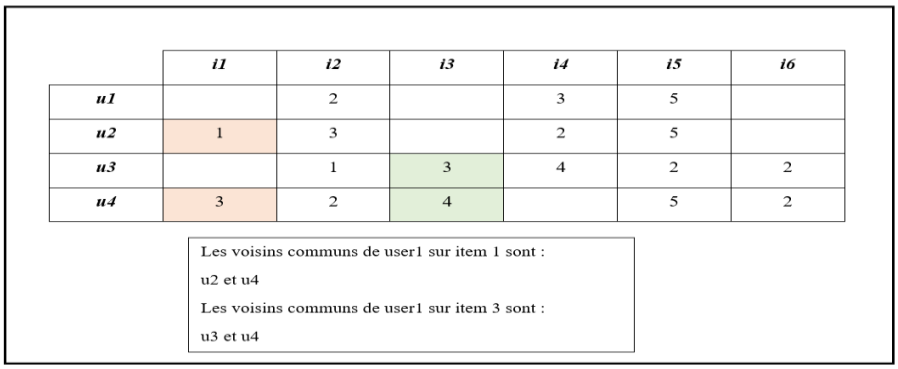


Figure 3.7 : *Résultat du calcul de CN_B.*

3.3.3.2.2. Coefficient de Jaccard

Pour chaque utilisateur cible u et chacun de ses items candidats i, nous proposons de calculer le coefficient de Jaccard $JC_B(u,i)$ comme suit :

$$JC_B = \frac{CN_B(u,i)}{|\{u' \in U | I_{u'} \cap I_u \neq \emptyset\}|} \quad (3.3)$$

Où :

- $CN_B(u,i)$, est le nombre de voisins communs entre u et i calculé par la formule (3.2),
- U, est l'ensemble de tous les utilisateurs,
- $I_{u'}$, I_u, sont les ensembles des items notés par les utilisateurs u' et u respectivement.

Exemple illustratif (suite)

- Le résultat du calcul de JC_B pour les utilisateurs : u_1, u_2, u_3 *et* u_4 ainsi que leurs items candidats est donné dans la figure suivante :

	i1	*i2*	*i3*	*i4*	*i5*	*i6*
u1	0.66	2	0.66	3	5	
u2	1	3	1	2	5	
u3	0.66	1	3	4	2	2
u4	3	2	4	1	5	2

Le coefficient Jaccard de u1 sur item 1 est : 0.66
Le coefficient Jaccard de u1 sur item 3 est : 0.66
Le coefficient Jaccard de user4 sur item 4 est : 1
Le coefficient Jaccard de user2 sur item 3 est : 0.66
Le coefficient Jaccard de user3 sur item 2 est : 1
Le coefficient Jaccard de user3 sur item 1 est : 0.66

Figure 3.8 : *Résultat du calcul de* JC_B.

3.3.4. Phase 4 : La recommandation

La tâche de prédiction de liens est communément connue comme une tâche de classification qui prédit l'existence d'un lien dans le futur. Cependant, le problème des systèmes de recommandation est modélisé comme une tâche de régression où une note (évaluation) est prédite et les meilleures notes prédites sont généralement recommandées à l'utilisateur.

Les mesures de similarité calculées dans la phase 3 trouvent un score pour chaque paire de nœud non liés dans le graphe biparti. Cette paire de nœud ne représente pas tout utilisateur et tout item qui ne sont pas liés mais elle a été filtrée et spécifiée lors des phases précédentes afin de devenir (utilisateur cible, item candidat).

Puisque ces scores ne donnent pas une note aux items, ceci nous amène à recommander les top-N items aux utilisateurs. Pour ce faire, nous allons suivre les étapes suivantes :

Etape 1 : Pour chaque utilisateur cible et ses items candidats, calculer les mesures CN_B et JC_B.

Etape 2 : Ordonner les scores trouvés par CN_B selon un ordre décroissant.

Etape 3 : Ordonner les scores trouvés par JC_B selon un ordre décroissant.

Etape 4 : Identifier la liste des N meilleurs scores.

Etape 5 : Recommander à l'utilisateur cible les items liés aux N meilleurs scores.

3.4. Conclusion

Dans ce chapitre nous avons présenté l'approche que nous avons proposée pour les systèmes de recommandation à base de prédiction de liens. Nous avons détaillé toutes les phases par lesquelles passe notre système : de l'exploration de données à la génération d'une liste de recommandation.

Nous présentons dans le chapitre suivant, les détails d'implémentation de notre application.

Chapitre 4 : Implémentation

4.1. Introduction

Dans ce chapitre, nous décrivons les travaux applicatifs et les détails de mise en œuvre de notre travail. Nous allons décrire d'abord la base de données utilisée pour valider notre système. Par la suite, nous présentons les outils et langage d'implémentation. Nous détaillons, ensuite, l'implémentation de nos étapes de travail, et nous terminons par les résultats obtenus et les expérimentations.

4.2. Base de données

4.2.1. Description de la base

Dans notre travail, nous avons utilisé la base de données des films MovieLens du groupe *GroupLens Research Center à l'Université du Minnesota* (Resnick et Varian 1997). Cette base est utilisée dans de nombreux projets de recherche liés aux systèmes de recommandation.

La base de données MovieLens est constituée de :

- ✓ 100,000 notes données sur une échelle de 1 à 5 (R).
- ✓ 943 utilisateurs (U) avec diverses informations : démographiques, personnels, ….
- ✓ 1682 (I) films avec des différentes informations (identifiant, titre, date de sortie, date de sortie de la vidéo, et 19 différents genres).
- ✓ Chaque utilisateur a noté au moins 20 films.
- ✓ Cette base est partagée en 80% des données en une base d'apprentissage et 20% en une base de test.

4.2.2. Echantillon de données

Pour effectuer nos expérimentations, nous avons choisi d'une manière aléatoire un échantillon de données de la base de données MovieLens. Cet échantillon est constitué de 453 notes *(R)* données par 86 utilisateurs *(U)* pour 50 films *(I)*. Cet échantillon reste toutefois extensible.

Le tableau suivant contient les informations concernant notre échantillon de données.

	Nombre d'utilisateurs	**Nombre d'items**	**Nombre de notes**
MovieLens Dataset	86	50	453

Tableau 4.1 : *Les valeurs correspondantes aux différents éléments utilisés dans notre échantillon de données.*

4.3. Mise en œuvre de la contribution

Dans cette section, nous allons spécifier, dans un premier temps, les outils utilisés pour développer notre application. Dans un second temps, nous allons décrire notre système de recommandation.

4.3.1. Outils et langage

- *Le langage Python*

Nous avons choisi le langage python comme langage de programmation. Il a été créé par Guido van Rossum. La première version de python est sortie en 1991. Python est un langage de programmation interprété, c'est-à-dire qu'il n'est pas nécessaire de le compiler avant de l'exécuter [2]. Python est gratuit, mais il peut être utilisé sans restriction dans des projets commerciaux. Il est portable, non seulement sur diverses variantes d'Unix, mais également sur des systèmes d'exploitation propriétaires : Mac OS, BeOS, NeXTStep, MS-DOS et diverses variantes de Windows. La syntaxe de Python est très simple, combinée à des types de données avancés (listes, dictionnaires...), permet de générer des programmes très compacts et très lisibles. Python est un langage évolutif, soutenu par une communauté d'utilisateurs enthousiastes et responsables, dont la plupart sont des partisans du logiciel libre (Swinnen 2012).

❖ *Le navigateur Anaconda*

Le navigateur Anaconda (figure 4.1) est une interface utilisateur graphique (GUI) de bureau inclus dans la distribution Anaconda qui nous permet de lancer des applications et de gérer facilement les packages, les environnements et les canaux conda sans utiliser des commandes de ligne de commande [3].

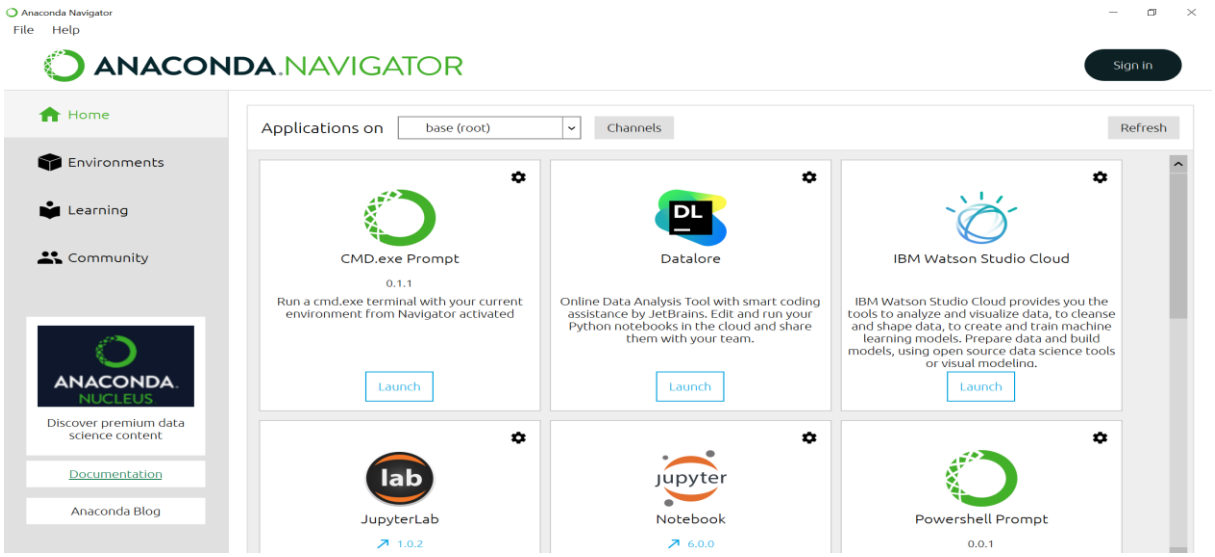


Figure 4.1 : *Navigateur Anaconda*

❖ *Jupyter NoteBook*

Jupyter Notebook (figure 4.2) et son interface flexible étendent le notebook au-delà du code à la visualisation, au multimédia, à la collaboration, etc. En plus d'exécuter le code, il stocke le code et la sortie, ainsi que les notes de démarque, dans un document modifiable appelé bloc-notes. Lorsqu'il est enregistré, celui-ci est envoyé du navigateur au serveur du bloc-notes, qui l'enregistre sur le disque en tant que fichier *JSON* avec une extension **.***ipynb* [4].

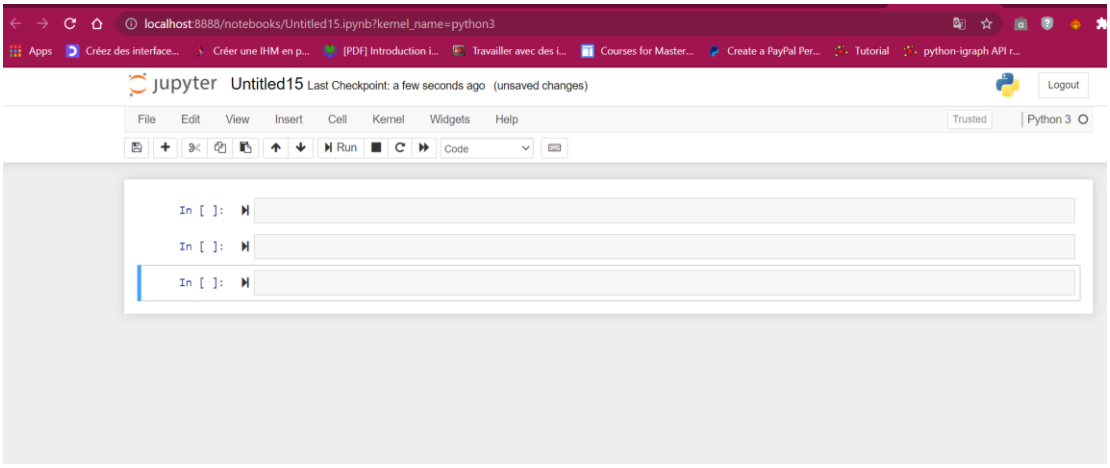


Figure 4.2 : *Jupyter NoteBook*

- *Package Igraph*

Igraph est une collection de bibliothèques pour créer et manipuler des graphiques et analyser des réseaux . Il est écrit en C et existe également sous forme de packages Python et R. igraph est capable de gérer efficacement de grands réseaux. Il peut être utilisé de manière productive avec un langage de programmation de haut niveau comme il peut supporter les utilisations interactives et non interactives [5].

- *Gephi*

Gephi est un logiciel libre d'analyse et de visualisation de réseaux, développé en Java et fondé sur la plateforme NetBeans. Gephi est notamment utilisé dans des projets de recherche scientifique et de journalisme de données. L'un des principaux intérêts de l'utilisation de Gephi pour cartographier des données est la possibilité d'utiliser de nombreux calculs liés à la théorie des graphes pour les appliquer aux données utilisées. Cela permet ainsi de visualiser quels sont les éléments d'un réseau les plus centraux, les plus éloignés, les mieux connectés, etc. De nombreux plugins existent également pour permettre de cartographier des ensembles de données très variés [6].

4.3.2. Description du système

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

Notre application est dotée d'une interface graphique GUI facile à utiliser qui permet la mise en œuvre des principaux composants de notre système.

L'interface principale de notre système est donnée par la figure 4.3.

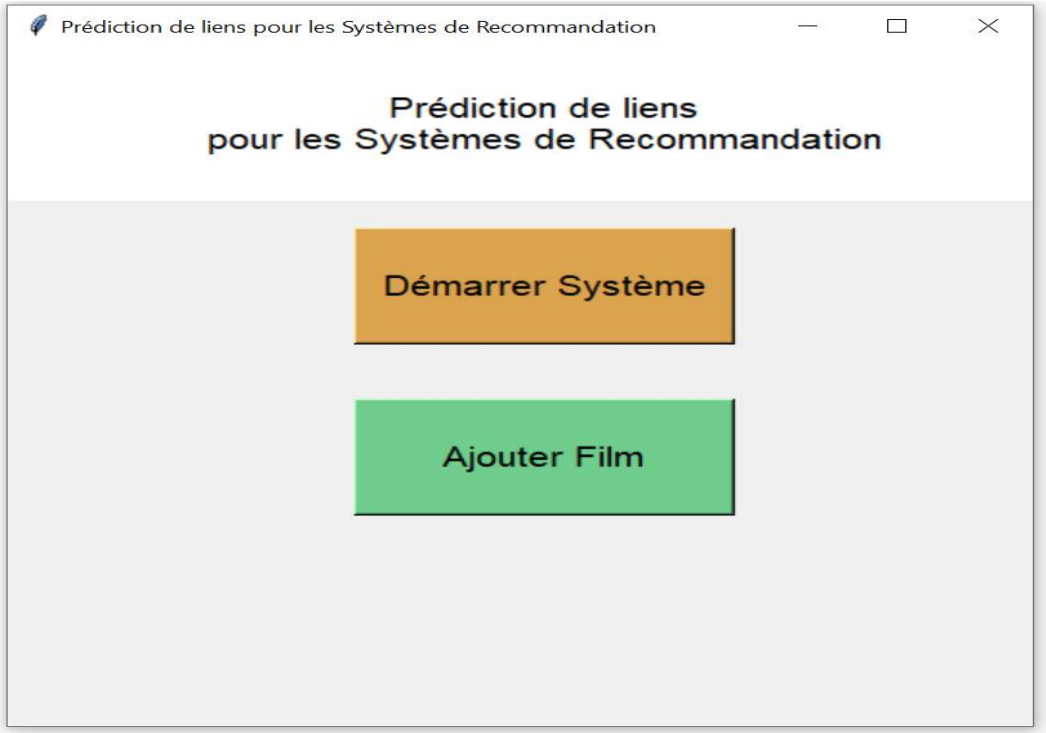
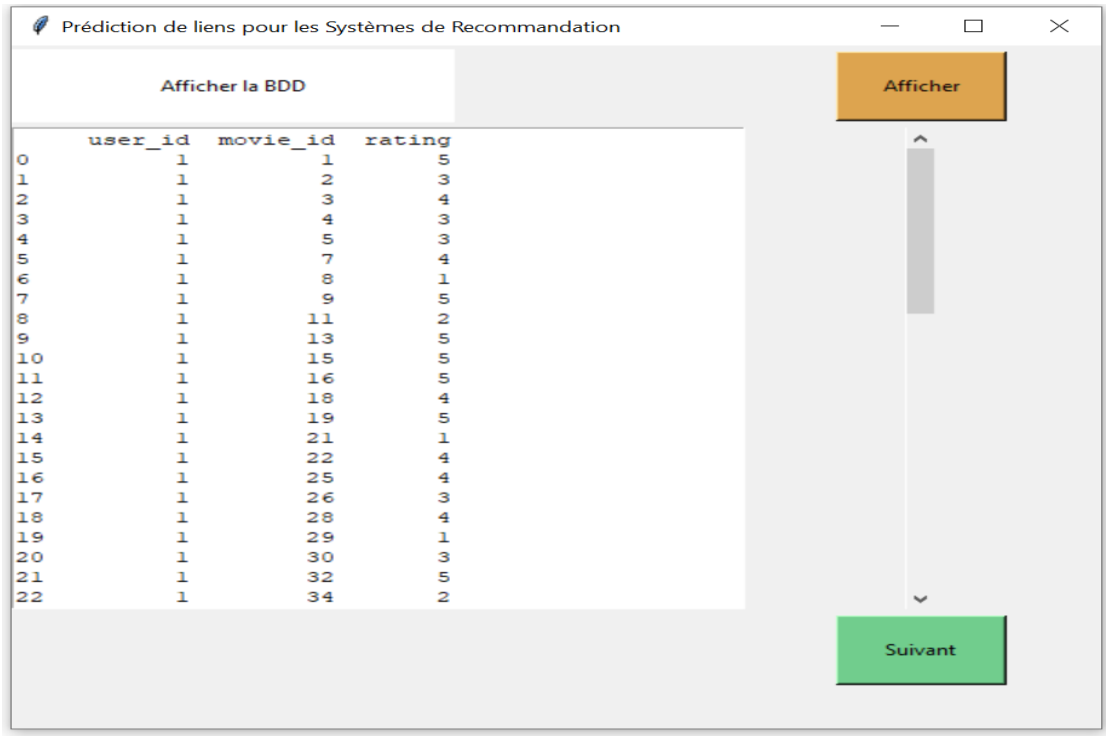


Figure 4.3 : *Interface d'accueil*

A travers cette page, l'utilisateur pourra accéder au système, comme il peut ajouter un nouvel item à la base de données.

Une fois l'utilisateur démarrera le système, l'interface de la figure 4.4 s'affiche. Dans cet écran, l'utilisateur peut afficher l'échantillon de données, après pré-traitement, utilisé pour valider notre système de recommandation. Le pré-traitement que nous avons réalisé sur la matrice de notes s'agit de l'élimination de la notion du temps d'attribution de la note. 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.

La figure suivante (figure 4.4) présente la matrice d'évaluation après le prétraitement.



	user_id	movie_id	rating
0	1	1	5
1	1	2	3
2	1	3	4
3	1	4	3
4	1	5	3
5	1	7	4
6	1	8	1
7	1	9	5
8	1	11	2
9	1	13	5
10	1	15	5
11	1	16	5
12	1	18	4
13	1	19	5
14	1	21	1
15	1	22	4
16	1	25	4
17	1	26	3
18	1	28	4
19	1	29	1
20	1	30	3
21	1	32	5
22	1	34	2

Figure 4.4 : *Notre échantillon de données après pré-traitement.*

Un deuxième pré-traitement a été réalisé sur les données relatives aux films. Nous avons filtré ces données pour ne garder que l'identifiant du film ainsi que ses genres qui sont nécessaires pour la construction du graphe de connaissance.

- ***Construction du graphe de connaissance des items***

Après avoir implémenté l'algorithme de génération du knowledge graphe présenté dans le chapitre précédant, nous avons obtenu un graphe mono-parti avec un seul type de nœuds qui sont les items (films), et un seul type de liaison : genres similaires. Le graphe généré contient 50 nœuds film et 728 liens (figure 4.5 et figure 4.6).

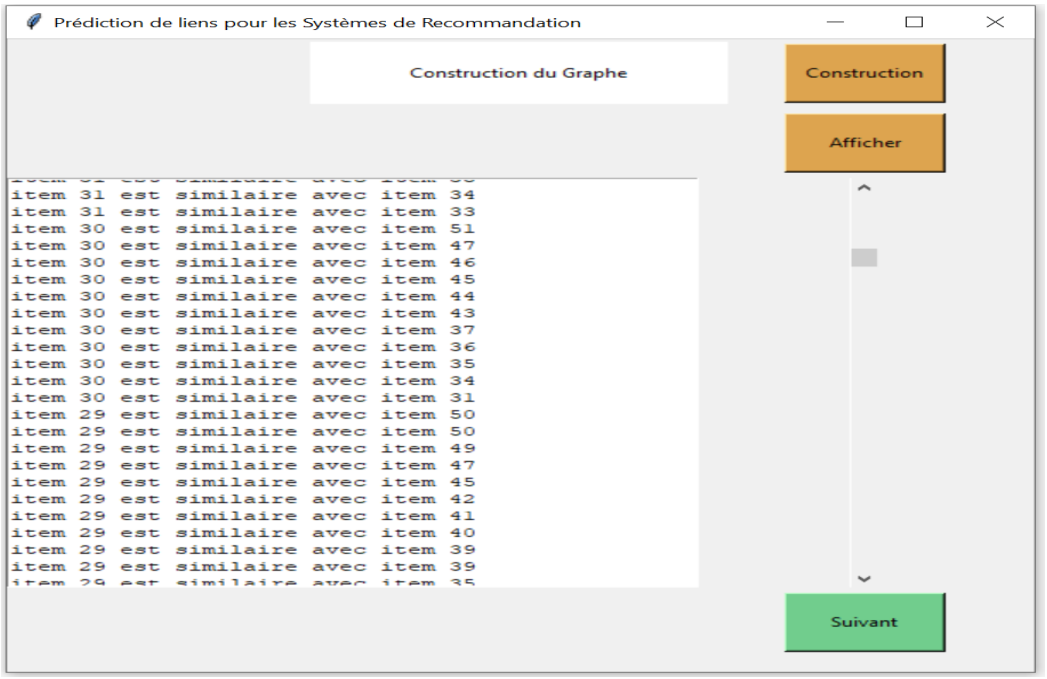


Figure 4.5 : *Construction de graphe des films.*

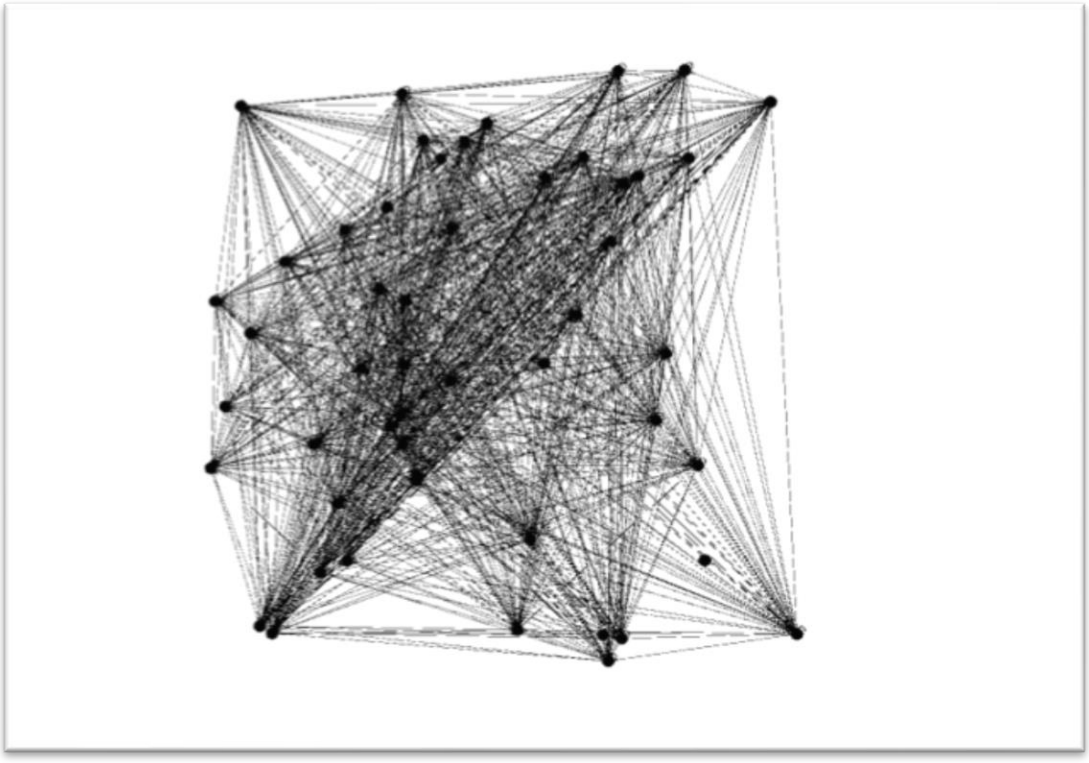


Figure 4.6 : *Le graphe de connaissance des films généré par Gephi.*

Comme nous l'avons présenté dans le chapitre précédent, sur ce graphe nous avons appliqué des algorithmes de détection de communautés afin d'obtenir des communautés des items.

- ***Détection de communautés***

A ce stade, nous avons tester et comparer les résultats de trois algorithmes de détection de communautés afin de ne garder que les communautés d'un seul algorithme pour la suite du travail. Les algorithmes testés sont : Infomap, l'algorithme de Newman et l'algorithme Louvain, ce choix a été justifié dans le chapitre précédent.

Nous avons tout d'abord créé un fichier *.gml* pour pouvoir utiliser le package *igraph*.

Les différentes communautés trouvées par les algorithmes : infomap, Newman et Louvain sont données dans les figures 3.7, 3.8 et 3.9 respectivement.

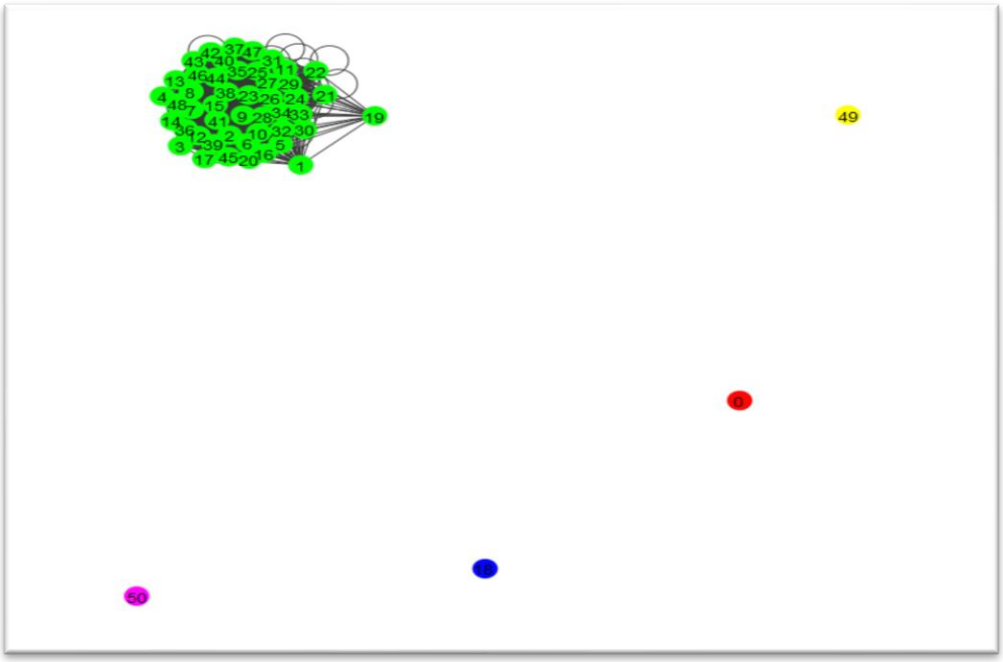


Figure 4.7 : *Communautés trouvées par Infomap.*

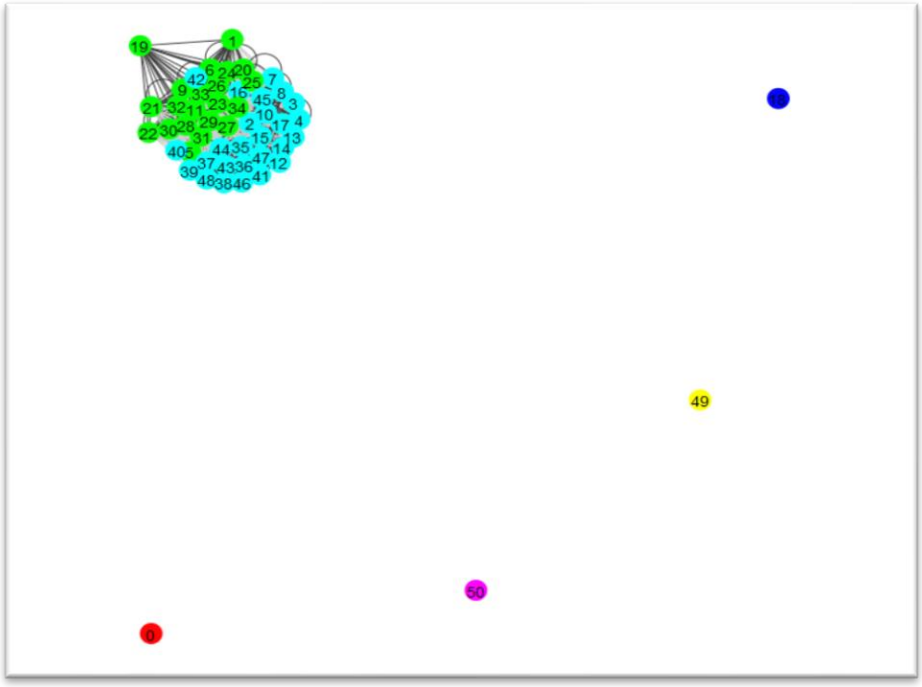


Figure 4.8 : *Communautés trouvées par Newman.*

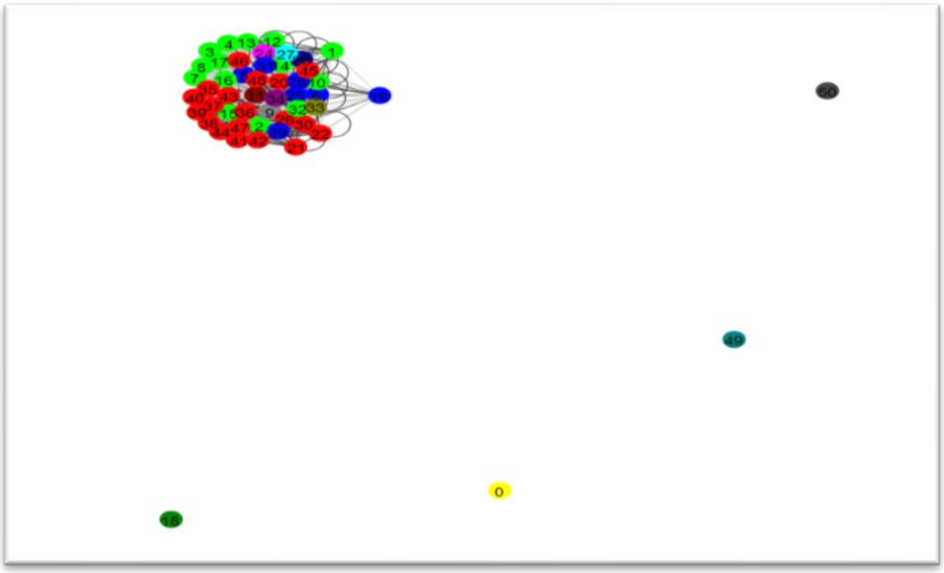


Figure 4.9 : *Communautés trouvées par Louvain.*

Ce qu'on peut constater de l'ensemble des communautés fournies par les différents algorithmes, c'est que seuls les algorithmes de Newman et Louvain nous ont donné plusieurs communautés diversifiées. L'algorithme infomap nous a fourni une communauté compacte contenant 47 nœuds

et chacun des nœuds restants forme une communauté à part. Afin d'assurer une diversité dans les items à recommander dans la phase suivante, nous avons décidé d'éliminer l'utilisation d'infomap.

Entre l'algorithme de Newman et Louvain, on s'est penchée vers la modularité comme critère de choix, puisque les deux algorithmes se basent sur la maximisation de la modularité lors de la génération des communautés. Le tableau 4.2 présentent les valeurs de la modularité trouvé par les deux algorithmes.

Algorithme	Newman	Louvain
Modularité	0.028	0.022

Tableau 4.2 : *Résultats de la modularité pour Newman et Louvain.*

Puisque l'algorithme de Newman nous offre une meilleure modularité, ce qui veut dire qu'il offre une meilleure qualité de communautés sur notre graphe, nous avons choisi de maintenir les communautés trouvées par cet algorithme pour la suite du travail (la phase de prédiction de liens).

- ***Prédiction de liens***

Après avoir obtenu les communautés des films dans l'étape précédente, nous avons utilisé cette information en concaténation avec les informations tirées de la matrice d'évaluation (notre échantillon de données) afin de déterminer pour chaque utilisateur la liste des items qui lui sont candidats pour la recommandation en suivant le processus proposé dans le chapitre précédent.

Par la suite, pour chaque couple (utilisateur cible, item candidat) nous avons calculé les mesures CN_B et JC_B donnée par les équations (3.2, 3.3) afin de prédire la possibilité de création de liens.

La figure ci-dessous présente l'interface de la phase de prédiction de liens (figure 4.10).

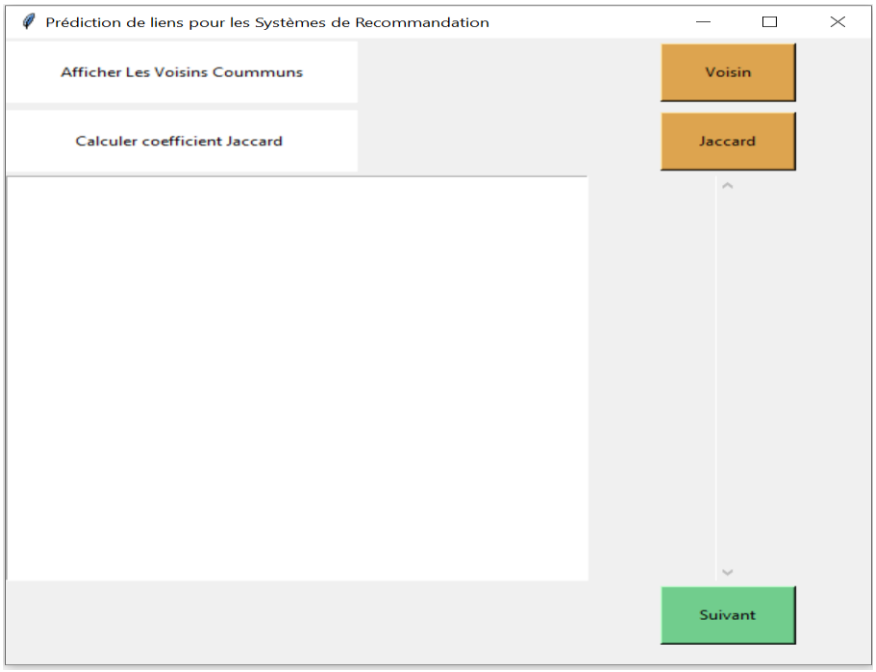


Figure 4.10 : *Interface de prédiction de liens.*

Les résultats des deux mesures adaptées de prédiction de liens est donné dans les figures 4.11 et 4.12.

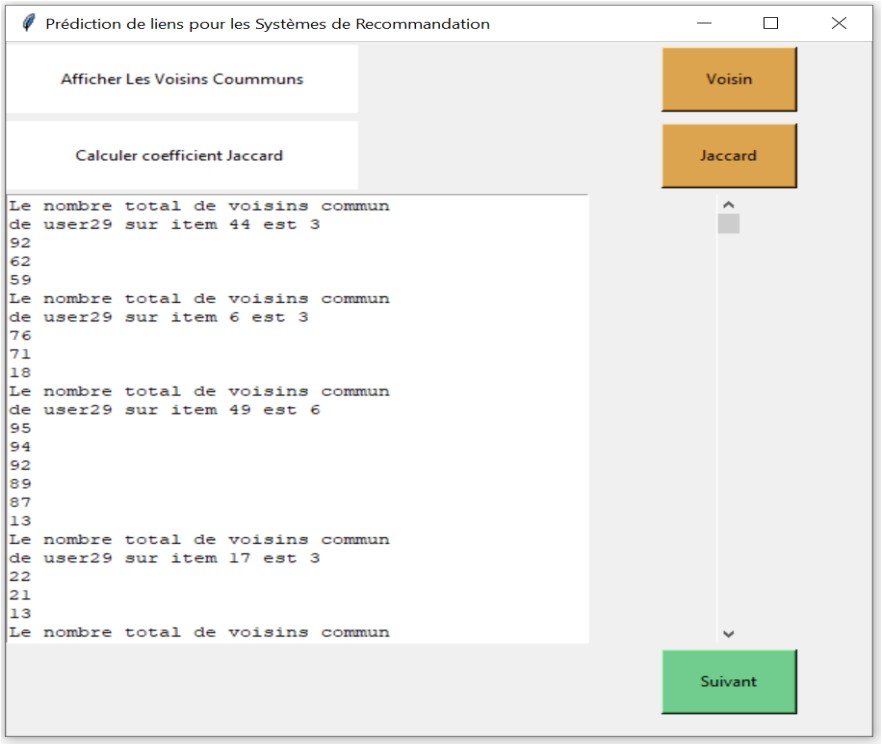


Figure 4.11 : *Résultats trouvés par* CN_B.

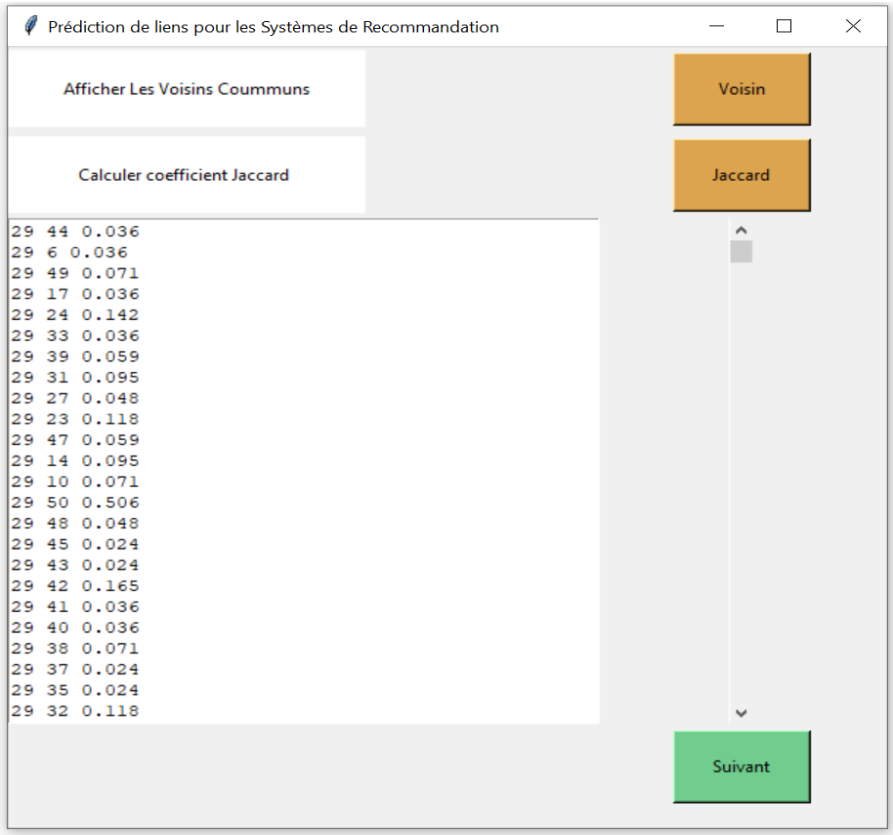


Figure 4.12 : *Résultats trouvés par* JC_B.

❖ ***La recommandation***

Après avoir calculé les deux mesures de prédiction de liens, il ne reste qu'à recommander à chaque utilisateurs les meilleurs items qui ne sont que les liens qui ont obtenus les meilleurs scores. Comme nous l'avons expliqué dans le chapitre précédent, nous avons opté pour une recommandation top-N. Par conséquent, nous avons ordonné les scores de liens trouvés et nous avons recommandé aux utilisateurs les top-10 films (les 10 premiers films dans le classement).

Les figures 4.13 et 4.14 présentent le résultat de la recommandation.

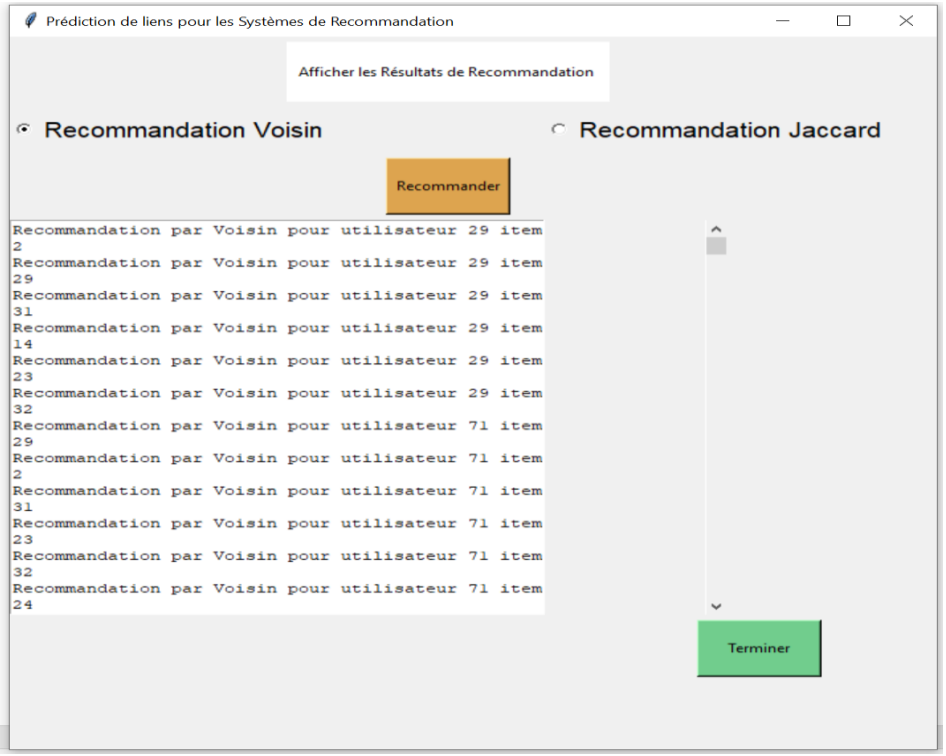


Figure 4.13 : *Recommandation basée sur* CN_B.

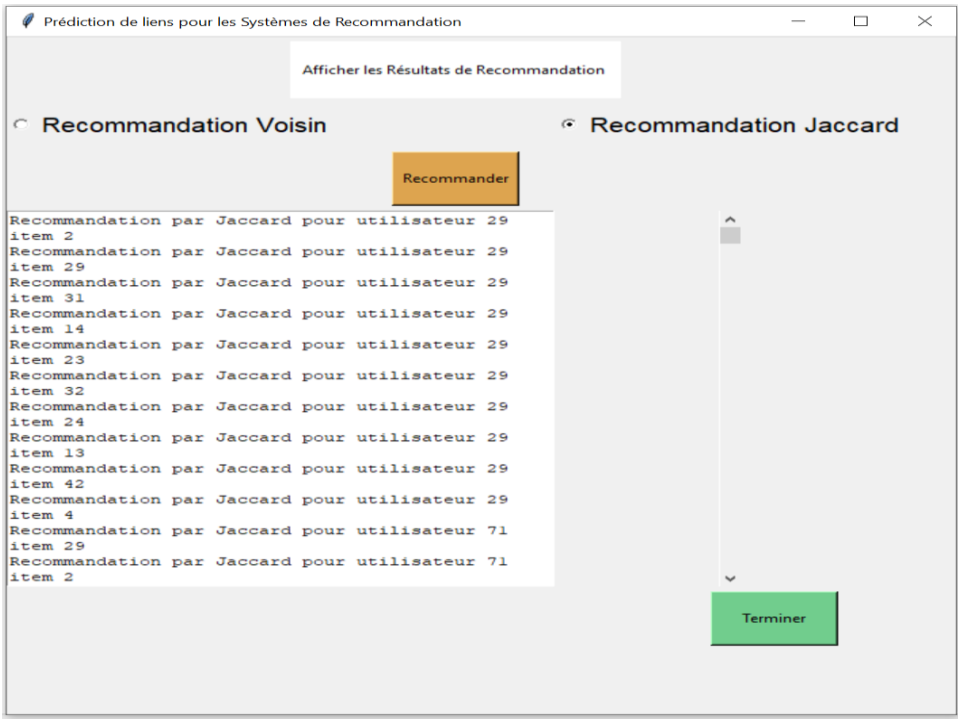


Figure 4.14 : *Recommandation basée sur* JC_B.

❖ **Ajout des films**

Notre système de recommandation prend en considération la possibilité d'intégrer un nouvel item. Ceci se fait en donnant un identifiant au film et en choisissant au moins un genre (figure 4.15).

Une fois l'action d'ajout est réalisée avec succès, ce film sera pris en considération lors des autres phases du système : construction du graphe de connaissance et détection de communautés.

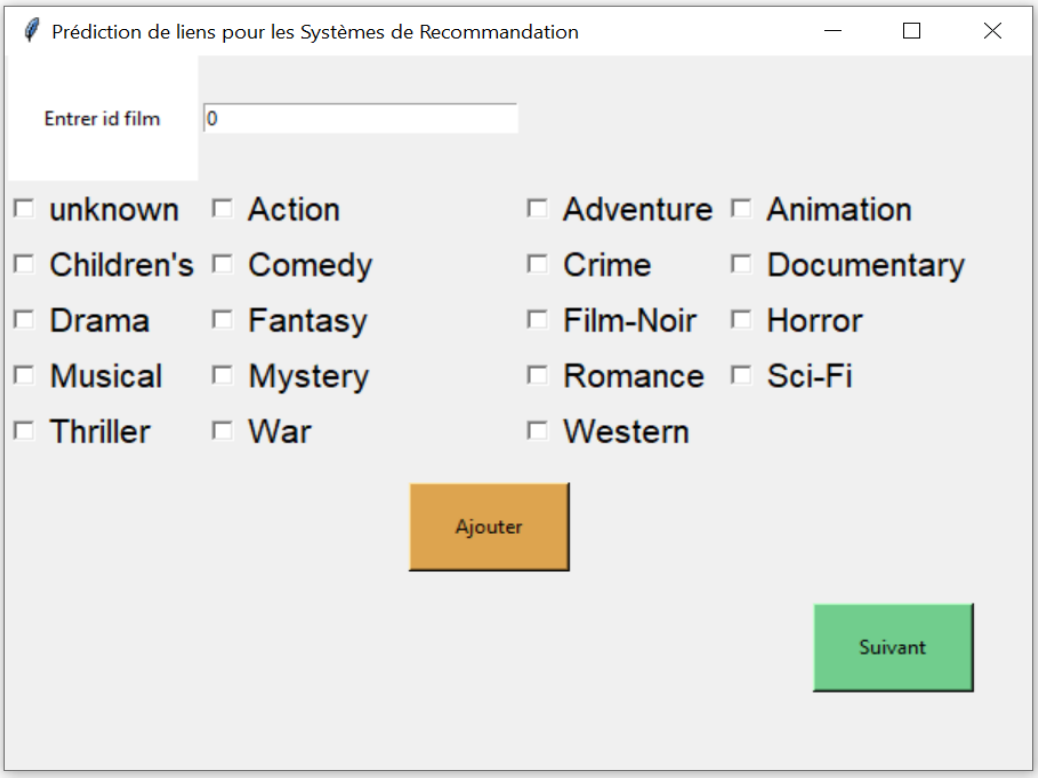


Figure 4.15 : *Interface d'ajout un nouvel item.*

4.4. Expérimentation

4.4.1. Evaluation

Nous allons, à présent, évaluer la performance de notre système de recommandation. Puisque la prédiction de liens fournit des scores liés à l'apparition de nouvelles liaisons et non pas des prédictions de notes, les mesures d'évaluation que nous avons présenté dans le chapitre 2 : MAE (équation 2.6), NMAE (équation 2.7), RMSE (équation 2.8) ne sont plus applicables dans notre

cas. Nous avons opté à l'utilisation des mesures issues de la classification à savoir, la précision, le rappel et la F-mesure.

- *Précision :* c'est une mesure d'exactitude qui mesure la proportion des items pertinents recommandés parmi tous les items recommandés. Autrement dit, la proportion des recommandations qui se sont avérées pertinentes.

$$Precision = \frac{Nombre\ d'items\ pertinents\ recommandés}{Nombre\ total\ des\ items\ recommandés} \quad (4.1)$$

- *Rappel :* c'est une mesure d'exhaustivité qui détermine la proportion des items pertinents recommandés parmi tous les items pertinents.

$$Rappel = \frac{Nombre\ d'items\ pertinents\ recommandés}{Nombre\ total\ d'itemspertinents} \quad (4.2)$$

- *F-mesure :* c'est la moyenne harmonique du rappel et de la précision.

$$\mathrm{F-mesure} = 2 \times \frac{\mathrm{Précision} \times \mathrm{Rappel}}{\mathrm{Précision} + \mathrm{Rappel}} \quad (4.3)$$

Pour pouvoir valider notre travail, nous avons extrait un ensemble de liaisons de notre échantillon de données c'est-à-dire un ensemble de triplets <utilisateur, item, note> afin de nous servir de base de test.

De plus, afin de pouvoir utiliser les trois mesures d'évaluation, nous devons transformer notre problème de recommandation en un problème binaire. Alors, nous avons divisé notre base de test en deux ensembles : des items pertinents et des items non pertinents en se basant sur leur note. Les items qui ont une note supérieure ou égale à 3 sont considérés comme pertinents et les items qui ont une note inférieure à 3 sont considérés comme non pertinents.

Le tableau 4.3 présente les valeurs des mesures d'évaluation : précision@10, rappel@10 et F-mesure@10 sur 15 utilisateurs de l'échantillon de données choisis au hasard.

Utilisateur	Précision@10	Rappel@10	F-mesure@10
49	0.5	1	0.67
83	0.81	1	0.9
43	1	1	1
7	0.7	1	0.83
12	0.7	1	0.83
18	0.81	1	0.9
94	0.91	1	0.96
28	1	1	1
9	0.91	1	0.96
17	0.81	0.89	0.85
23	0.5	1	0.67
24	0.6	1	0.75
32	0.91	0.82	0.87
37	1	1	1
45	0.7	0.88	0.78

Tableau 4.3 : *Résultats des mesures d'évaluation pour 15 utilisateurs.*

Le tableau 4.4 donne la moyenne de la précision@10, le rappel@10 et la F-mesure@10 sur tout l'échantillon de données, les 86 utilisateurs et 50 items.

	Précision@10	Rappel@10	F-mesure@10
Total échantillon	0.75	0.97	0.84

Tableau 4.4 : *La moyenne des mesures d'évaluation sur notre échantillon de données.*

La moyenne des trois mesures est donnée aussi dans la figure 4.16.

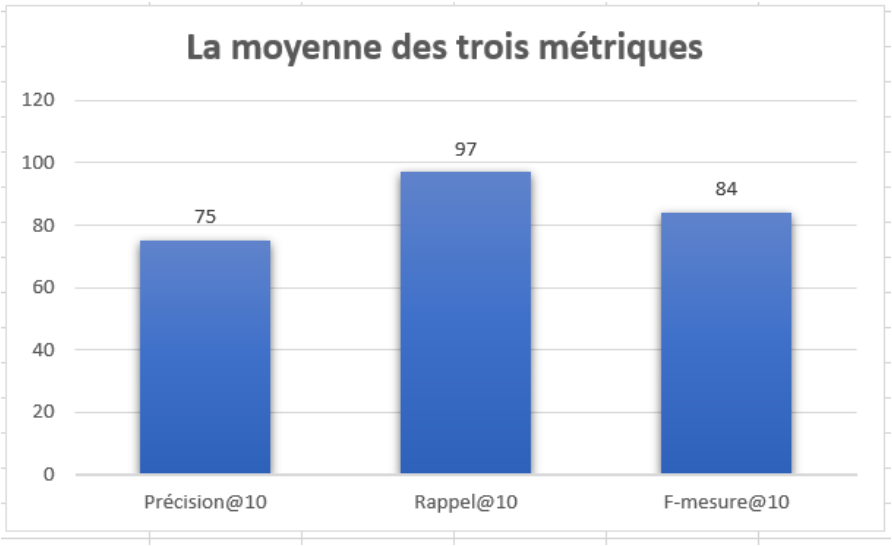


Figure 4.16 : *La moyenne des trois métriques d'évaluation sur les 86 utilisateurs de l'échantillon de données.*

4.4.2. Discussion

Après avoir validé notre système de recommandation, nous avons constaté que les valeurs des métriques d'évaluation sont satisfaisantes.

En effet, pour la précision@10, on trouve que pour les 10 utilisateurs choisis au hasard donné dans le tableau 4.3, sa valeur minimale est de 0,5 et sa valeur maximale est de 0,91. Idem pour tout l'échantillon de données, sa valeur minimale est de 0.5 et sa valeur maximale est de 1. Ceci dit que sur tout l'échantillon de données 75% des recommandations sont pertinentes pour l'utilisateur.

Pour le rappel@10, on trouve que pour les 10 utilisateurs choisis au hasard donné dans le tableau 4.3, sa valeur minimale est de 0,89 et sa valeur maximale est de 1. Idem pour tout l'échantillon de données, sa valeur minimale est de 0.75 et sa valeur maximale est de 1. Ceci dit que sur tout l'échantillon de données 97% du total des items pertinents apparaissent dans les 10 premiers résultats (top-10 recommandation).

Ces résultats nous permettent de conclure que notre système de recommandation à base de graphe, à travers les différentes phases que nous avons proposées et l'adaptation de prédiction de liens que nous avons réalisée, réussit à recommander aux utilisateurs des items qui correspondent à leurs préférences.

4.5. Conclusion

Dans ce chapitre, nous avons présenté les outils de développement et les étapes d'implémentation et les différentes interfaces de notre application. Nous avons également présenté les résultats des évaluations exprimés en matière de précision, de rappel et de F-mesure @10. Finalement, nous avons discuté les résultats obtenus.

Le but principal de cette implémentation est d'évaluer notre proposition de système de recommandation à base de prédiction de liens et de valider et justifier les différentes phases proposées afin d'améliorer la qualité de recommandation.

Conclusion générale

Les Systèmes de Recommandation (SR) sont un outil de recherche d'information et de filtrage qui vise à proposer aux utilisateurs des items qui pourraient les intéresser. La plupart des SR se basent sur l'analyse d'historique d'évaluation des items par les utilisateurs afin de prédire l'intérêt qu'un utilisateur peut porter à un item donné.

L'historique d'évaluation est souvent représenté sous forme d'une matrice $R : U \times I$ où un élément de cette matrice représente l'évaluation (note) qu'un utilisateur donne à un item. L'objectif de la recommandation est, alors, de prédire les valeurs manquantes dans cette matrice. Les techniques traditionnelles de système de recommandation souffrent de certains problèmes relatifs à cette matrice. Dans le cadre de ce projet, on s'intéresse à l'utilisation des approches de prédiction de liens pour le calcul de recommandation afin de faire face aux manques de données.

Nous nous intéressons, dans notre travail, à la prédiction de liens dans un graphe *utilisateur-item* en vue de la recommandation. Nous avons, dans un premier temps, exploré les corrélations existantes entre les différents items afin de construire un premier graphe mono-parti indépendant des utilisateurs et de leurs appréciations. Ceci nous a facilité l'intégration d'un nouvel item comme ça nous a fourni des informations utiles qui ont guidé la phase de prédiction de liens.

Dans un second temps, nous avons procédé au calcul de la prédiction de liens entre les utilisateurs et les items. Nous avons adapté les métriques de proximité : CN et JC afin de prendre en considération les spécificités de notre graphe. A la fin, une liste des top-10 meilleurs recommandations a été fournie à chaque utilisateur.

Les résultats que nous avons obtenus (en termes de précision, de rappel et de F-mesure) sont encourageants et nous permettent de conclure que notre système de recommandation à base de graphe, à travers les différentes phases que nous avons proposées et l'adaptation de prédiction de liens que nous avons réalisée, réussit à recommander aux utilisateurs des items qui correspondent à leurs préférences.

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

A cours terme, nous envisageons, tout d'abord, de tester la scalability de notre système en l'appliquant sur la totalité de la BDD MovieLens. L'idée d'intégrer la détection de communautés sur le graphe des items a été bénéfique sur le plan informationnel pour la phase de prédiction de liens. Ceci nous conduit à envisager de tirer plus de profit des informations fournies par cette technique et de considérer les communautés des items lors du calcul de la prédiction de liens. Une autre piste que nous voulons explorer dans le futur proche est d'exploiter les notes données aux items afin de construire un graphe pondéré.

A long terme, nous envisageons de considérer l'aspect temporel de la BDD et ainsi intégrer la notion du temps dans le graphe ceci nous permettra de valider la prédiction de liens sur une base de test réelle.

Références bibliographiques

Adali, Sibel, Sisenda, Fred, et Magdon-Ismail, Malik. 2012. «Actions speak as loud as words: Predicting relationships from social behavior data». In *Proceedings of the 21st international conference on World Wide Web* (pp. 689-698).

Adamic, Lada. A., et Adar, Eytan. 2003. «Friends and neighbors on the web». *Social networks*, *25*(3), 211-230.

Adomavicius, Gediminas, et Tuzhilin, Alexander. 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.

Ahmed, Nahla Mohamed, et Ling Chen. 2016. «An efficient algorithm for link prediction in temporal uncertain social networks.» *Information Sciences 331* 120-136.

Ai, Jun, Liu, Yayun, Su, Zhan, Zhang, Hui, et Zhao, Fengyu. 2019. «Link prediction in recommender systems based on multi-factor network modeling and community detection». *EPL (Europhysics Letters)*, *126*(3), 38003.

Ajay, Kumar, Singh Shashank Sheshar, Singh Kuldeep, et Biswas Bhaskar. 2020. «Link prediction techniques, applications, and performance: A survey.» *Physica A: Statistical Mechanics and its Applications, Elsevier* 553, 124289.

Backstrom, Lars, et Jure Leskovec. 2011. «Supervised Random Walks: Predicting and Recommending Links in Social Networks.» *the 4th ACM International Conference on Web Search and Data Mining (WSDM'11).*

Béchet, Nicolas. 2012. État de l'art sur les Systèmes de Recommandation. *Projet AxIS de l'INRIA, dans le cadre du projet Addictrip*.

Bedi, Punam et Sharma, Chhavi. 2016. «Community detection in social networks». *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *6*(3), 115-135.

Benchettara, Nesrine, Kanawati, Rushed, et Rouveirol, Céline. 2010. «A supervised machine learning link prediction approach for academic collaboration recommendation». In *Proceedings of the fourth ACM conference on Recommender systems* (pp. 253-256).

Benchettara, Nesrine, Kanawati, Rushed et Rouveirol, Céline. 2010. «Supervised machine learning applied to link prediction in bipartite social networks». In *2010 international conference on advances in social networks analysis and mining* (pp. 326-330). IEEE.

Bhawsar, Y., et G. Thakur. 2016. «Performance Evaluation of Link Prediction Techniques Based on Fuzzy Soft Set and Markov Model.» *Taylor Francis Online* 113-126. doi:https://doi.org/10.1016/j.fiae.2016.03.007.

Bhuyan, Monowar, H., Bhattacharyya, D. K., et Kalita, J. K. 2016. «A multi-step outlier-based anomaly detection approach to network-wide traffic». *Information Sciences*, *348*, 243-271.

Blondel, Vincent D., Guillaume, Jean-Loup L., Lambiotte, Renault, et Lefebvre, E. 2008. «Fast unfolding of communities in large networks». *Journal of statistical mechanics: theory and experiment*, *2008*(10), P10008.

Bobadilla, J., Ortega, F., Hernando, A., et Gutiérrez, A. 2013. «Recommender systems survey». *Knowledge-based systems*, *46*, 109-132.

Burke, R. 2002. «Hybrid recommender systems: Survey and experiments». *User modeling and user-adapted interaction*, *12*(4), 331-370.

Burke, R. 2007. «Hybrid web recommender systems». *The adaptive web*, 377-408.

Canu, Mael. 2017. «*Détection de communautés orientée sommet pour des réseaux mobiles opportunistes sociaux*». Doctoral dissertation, Université Pierre et Marie Curie-Paris VI.

Chuan, Pham. Minh, Ali, Mumtaz., Khang, Tran, Dinh, et Dey, Nilanjan. 2018. Link prediction in co-authorship networks based on hybrid content similarity metric. *Applied Intelligence*, *48*(8), 2470-2486.

Clauset, Aaron, Moore, Christopher, et Newman, M. E. 2008. «Hierarchical structure and the prediction of missing links in networks». *Nature*, *453*(7191), 98-101.

Cui, Y., Liu, Y., Hu, J., et Li, H. 2018. «A survey of link prediction in information networks». In *2018 IEEE International Conference on Smart Internet of Things (SmartIoT)* (pp. 29-33). IEEE.

Daud, N. N., Ab Hamid, S. H., Saadoon, M., Sahran, F., et Anuar, N. B. 2020. «Applications of link prediction in social networks: A review». *Journal of Network and Computer Applications*, *166*, 102716.

El-Moussaoui, Mohamed, Tarik Agouti, Abdessadek Tikniouine, et Abdessadek Tikniouine. 2019. «A comprehensive literature review on community detection: Approaches and applications.» *Procedia Computer Science* 151, 295-302.

Farashah, Mohammadsadegh Vahidi, Akbar Etebarian, Reza Azmi, et Ebrahimzadeh Dastjerdi Reza. 2021. «A Hybrid Recommender System Based-on Link Prediction for Movie Baskets Analysis.» *Journal of Big Data* 32.

Fortunato, Santo. 2010. «Community detection in graphs.» *Physics Reports* 75-174.

Fortunato, Santo, et Darko Hric. 2016. «Community detection in networks: A user guide.» *Physics Reports* 1-43.

Fouss, F., Pirotte, A., Renders, J. M., et Saerens, M. 2007. «Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation». *IEEE Transactions on knowledge and data engineering*, *19*(3), 355-369.

Fu, C. H., Chang, C. S., et Lee, D. S. 2014. «A proximity measure for link prediction in social user-item networks». In *Proceedings of the 2014 IEEE 15th International Conference on Information Reuse and Integration (IEEE IRI 2014)* (pp. 710-717). IEEE.

Garg, Amit, Senthilnathan Viswanathan, et Shloka Desai. 2015. *CS224W Project Final Report: Using community detection and link prediction to improve Amazon recommendations.* Stanford University: Stanford University.

Getoor, L., et Taskar, B. 2007. Statistical relational learning.

Gong, N. Z., Talwalkar, A., Mackey, L., Huang, L., Shin, E. C. R., Stefanov, E., et Song, D. 2014. «Joint link prediction and attribute inference using a social-attribute network». *ACM Transactions on Intelligent Systems and Technology (TIST), 5*(2), 1-20.

Guimerà, R., et Sales-Pardo, M. 2009. «Missing and spurious interactions and the reconstruction of complex networks». *Proceedings of the National Academy of Sciences*, *106*(52), 22073-22078.

Al Hasan, M., et Zaki, M. J. 2011. «A survey of link prediction in social networks». In *Social network data analytics* (pp. 243-275). Springer, Boston, MA.

Heckerman, David, David Maxwell Chickering, Christopher Meek, Robert Rounthwaite, et Kadie Carl. 2001. «Dependency Networks for Inference, Collaborative Filtering, and Data Visualization.» *Journal of Machine Learning Research.*

Herlocker, J. L., Konstan, J. A., Terveen, L. G., et Riedl, J. T. 2004. «Evaluating collaborative filtering recommender systems». *ACM Transactions on Information Systems (TOIS), 22*(1), 5-53.

Huang, Keman, Yushun Fan, Wei Tan, et Xiang Li. 2013. «Service Recommendation in an Evolving Ecosystem: A Link Prediction Approach.» *IEEE 20th International Conference on Web Services.* Santa Clara, CA, USA: IEEE. 8.

Huang, Z. 2010. Link prediction based on graph topology: The predictive value of generalized clustering coefficient. *Available at SSRN 1634014*.

Kart, Ozge, Oguzhan Ulucay Ulucay, Berkay Bingol, et Zerrin Isik. 2020. «A machine learning-based recommendation model for bipartite networks.» *Physica A* 8.

Kaya, B. 2020. «A hotel recommendation system based on customer location: a link prediction approach». *Multimedia Tools and Applications*, *79*(3), 1745-1758.

Kaya, B. 2020. «Hotel recommendation system by bipartite networks and link prediction». *Journal of Information Science*, *46*(1), 53-63.

Kaya, B., et Poyraz, M. 2015. Age-series based link prediction in evolving disease networks. *Computers in biology and medicine*, *63*, 1-10.

Kohavi, R., Longbotham, R., Sommerfield, D., et Henne, R. M. 2009. «Controlled experiments on the web: survey and practical guide». *Data mining and knowledge discovery*, *18*(1), 140-181.

Koptelov, Maksim, Albrecht Zimmermann, Bruno Crémilleux, et Lina Soualmia. March 2020. «Link prediction via community detection in bipartite multi-layer graphs.» *Proceedings of the 35th Annual ACM Symposium on Applied Computing.* 430-439.

Koptelov, Maksim, Albrecht Zimmermann, et Bruno Cremilleux. 2019. «Link Prediction via Community Detection in Bipartite Multi-Layer Graphs.» *Workshop GEM: Graph Embedding and Mining co-located with ECML/PKDD.*

Kou, Huaizhen, Hanwen Liu, Yucong Duan, Wenwen Gong, Yanwei Xu, Xiaolong Xu, et Lianyong Qi. 2020. «Building trust/distrust relationships on signed social service network through privacy-aware link prediction process.» *Applied Soft Computing* 10.

Kovács, I. A., Luck, K., Spirohn, K., Wang, Y., Pollis, C., Schlabach, S., et Barabási, A. L. 2019. Network-based prediction of protein interactions. *Nature communications*, *10*(1), 1-8.

Kunegis, J., De Luca, E. W., et Albayrak, S. 2010. «The link prediction problem in bipartite networks». In *International Conference on Information Processing and Management of Uncertainty in Knowledge-based Systems* (pp. 380-389). Springer, Berlin, Heidelberg.

Kuo, T. T., Yan, R., Huang, Y. Y., Kung, P. H., et Lin, S. D. 2013. «Unsupervised link prediction using aggregative statistics on heterogeneous social networks». In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 775-783).

Kuo, Tsung-Ting, Rui Yan, Yu-Yang Huang, Perng-Hwa Kung, et Shou-De Lin. 2013. «Unsupervised Link Prediction Using Aggregative Statistics on Heterogeneous Social Networks.» *the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD13).* Chicago, Illinois, USA: ACM Digital Library. 9.

Lakshmi, T. J., et Bhavani, S. D. 2021. Link Prediction Approach to Recommender Systems. *arXiv preprint arXiv:2102.09185*.

Lakshmi, T. Jaya, et S. Durga Bhavani. 2018. «Link prediction measures in various types of information networks: a review.» *Proceedings of the 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining.* IEEE. 1160-1167.

Li, Jing, Lingling Zhang, Fan Meng , et Fenhua Li. 2014. «Recommendation Algorithm Based On Link Prediction And Domain Knowledge In Retail Transactions.» *2nd International Conference on Information Technology and Quantitative Management, ITQM.* Beijing 100190, China: Procedia Computer Science 31. 875 – 881.

Li, Zhepeng, Xiao Fang, et Olivia Sheng. 2017. «A Survey of Link Recommendation for Social Networks: Methods,Theoretical Foundations, and Future Research Directions.» *ACM Transactions on Management Information Systems (TMIS)* 1-26.

Liben-Nowell, D., et Kleinberg, J. 2007. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, *58*(7), 1019-1031.

Liu, W., et Lü, L. (2010). «Link prediction based on local random walk». *EPL (Europhysics Letters)*, *89*(5), 58007.

Liu, Weiping, et Linyuan Lü. 2010. «Link prediction based on local random walk.» *Europhysics Letters Association.*

Lu, Linyuan, et Zhou Tao. 2010. «Link Prediction in Complex Networks: A Survey.» *Elsevier Science* 44.

Maatallah, Majda, 2016. *Une Technique Hybride pour les Systèmes de Recommandation.* (Doctoral dissertation, Université Badji Mokhtar Annaba).

Mathieu, Vincent. 2001. *Outils d'analyse statique*. Université Laval, Dép. d'informatique.

Matias, Catherine. 2015. Notes de cours : Analyse statistique de graphes. Université Pierre et Marie et Curie.

Mohamed, Marwa Hussien, Mohamed Helmy Khafagy, et Mohamed Hasan Ibrahim. 2019. «Recommender Systems Challenges and Solutions A Survey.» *2019 International Conference on Innovative Trends in Computer Engineering (ITCE'2019).* Aswan, Egypt: IEEE.

Newman, M. E. 2006. «Finding community structure in networks using the eigenvectors of matrices». *Physical review E*, *74*(3), 036104.

Prajapati, K., Shah, H., et Mehta, R. 2020. «A Survey Of Link Prediction In Social Network Using Deep Learning Approach». International journal of scientific & technology research.

Resnick, Paul, et Hal R. Varian. 1997. «Recommender systems.» *Communication of the ACM* 40, 56-58.

Rosvall, M., et Bergstrom, C. T. (2008). «Maps of random walks on complex networks reveal community structure». *Proceedings of the national academy of sciences*, *105*(4), 1118-1123.

Schlichtkrull, M., T.N. Kipf, P. Bloem, R. van den Berg, I. Titov, et M. Welling. 2018. «Modeling Relational Data with Graph Convolutional Networks.» *The Semantic Web, Springer International Publishing* pp. 593–607.

Shani, G., & Gunawardana, A. 2011. Evaluating recommendation systems. In *Recommender systems handbook* (pp. 257-297). Springer, Boston, MA.

Shen, Dou, Jian-Tao Sun, Qiang Yang, et Zheng Chen. 2006. «Latent Friend Mining from Blog Data.» *Sixth International Conference on Data Mining (ICDM'06).* IEEE.

Sigward, Eric. 2002. *Introduction à la théorie des graphes.* Université Paris.

Silveira, Thiago, Min Zhang, Xiao Lin, Yiqun Liu, et Shaoping Ma. 2019. «How good your recommender system is? A survey on evaluations in recommendation.» *International Journal of Machine Learning and Cybernetics* .

Slokom, M., et Ayachi, R. 2017. «A new social recommender system based on link prediction across heterogeneous networks». In *International Conference on Intelligent Decision Technologies* (pp. 330-340). Springer, Cham.

Srilatha, Pulipati, et Manjula Ramakrishnan. 2016. «Similarity Index based Link Prediction Algorithms in Social Networks : A Survey.» *Journal of Telecommunications And Information Technology* 8.

Su, Z., Zheng, X., Ai, J., Shang, L., et Shen, Y. 2019. «Link prediction in recommender systems with confidence measures». *Chaos: An Interdisciplinary Journal of Nonlinear Science*, *29*(8), 083133.

Su, Zhan, Xiliang Zheng, Jun Ai, Yuming Shen, et Xuanxiong Zhang. 2020. «Link prediction in recommender systems based on vector similarity.» *Physica A* 12.

Swinnen, Gérard. 2012. *Apprendre à programmer avec Python 3.* Broché – Livre grand format, février 2012. Eyrolles.

Tadlaoui, Mohammed. 2018. *Système de recommandation de ressources pédagogiques fondé sur les liens sociaux : formalisation et évaluation.* THESE de DOCTORAT DE L'UNIVERSITE DE LYON opérée au sein de L'INSA Lyon et délivrée en partenariat international avec L'université de Tlemcen.

Talasu, N., Jonnalagadda, A., Pillai, S. S. A., et Rahul, J. 2017. «A link prediction based approach for recommendation systems». In *2017 international conference on advances in computing, communications and informatics (ICACCI)* (pp. 2059-2062). IEEE.

Tong, Hanghang, Christos Faloutsos, et Jia-Yu Pan. 2006. «Fast Random Walk with Restart and Its Applications.» *the 6th IEEE International Conference on Data Mining (ICDM'06).*

Wang, P., Xu, B., Wu, Y., et Zhou, X. 2015. «Link prediction in social networks: the state-of-the-art». *Science China Information Sciences*, *58*(1), 1-38.

Wang, Chao, Venu Satuluri, et Srinivasan Parthasarathy. 2007. «Local Probabilistic Models for Link Prediction.» *7th IEEE International Conference on Data Mining (ICDM07).* 10.

Wang, Dashun, Dino Pedreschi, Chaoming Song, Fosca Giannotti, et Albert László Barabási. 2011. «Human Mobility, Social Ties, and Link Prediction.» *the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD11).*

Wang, H., et Le, Z. 2020. «Seven-Layer Model in Complex Networks Link Prediction: A Survey». *Sensors*, *20*(22), 6560.

Wang, H., Le, Z., et Gong, X. 2020. «Recommendation System Based on Heterogeneous Feature: A Survey». *IEEE Access*, *8*, 170779-170793.

Wang, Quan, Zhendong Mao, Bin Wang, et Li Guo. 2017. «Knowledge Graph Embedding: A Survey of Approaches and Applications.» *IEEE Transactions on Knowledge and Data Engineering* 2723-2743.

Xie, F., Chen, Z., Shang, J., Feng, X., et Li, J. 2015. «A link prediction approach for item recommendation with complex number». *Knowledge-Based Systems*, *81*, 148-158.

You, J., Liu, B., Ying, R., Pande, V., et Leskovec, J. 2018. «Graph convolutional policy network for goal-directed molecular graph generation». *arXiv preprint arXiv:1806.02473*.

Zhang, Le, et P.N. Suganthan. 2016. «A survey of randomized algorithms for training neural networks.» Dans *Information Sciences 364-365*, de Le Zhang, 146-155. ELSEVIER.

Zhou, Tao, Linyuan Lu, et Yi-Cheng Zhang. 2009. «Predicting Missing Links via Local Information.» *Eur. Phys. J. B 71*.

Webographie

[1]https://towardsdatascience.com/various-implementations-of-collaborative-filtering-100385c6dfe0 consulté le : 20/05/2021

[2] https://python.doctor consulté le : 03/09/2021

[3] https://docs.anaconda.com/anaconda/navigator/ consulté le : 03/09/2021

[4] https://jupyter.readthedocs.io/en/latest/projects/architecture/content-architecture.html consulté le : 03/09/2021

[5] https://igraph.org/python/doc/tutorial/install.html consulté le : 03/09/2021

[6] https://enexdi.sciencesconf.org/data/pages/GEPHI_TUTORIEL.pdf consulté le 03/09/2021