الجمهورية الجزائرية الديمقراطية الشعبية

وزارة التعليم العالي والبحث العلمي

UNIVERSITÉ BADJI MOKHTAR - ANNABA
BADJI MOKHTAR – ANNABA UNIVERSITY

جامعة باجي مختار – عنابـــة

Faculté : séances de l'Ingéniorat

Département : Informatique

Domaine : Mathématique Informatique

Filière : Informatique

Spécialité : Gestion et analyse des données massives

Mémoire

Présenté en vue de l'obtention du Diplôme de Master

Thème:

La création d'un système de recommandation explicatif bas ésur les Tags.

Présenté par : *Hemza Zarzouni*

Encadrant: Dr. Mendjel Med Said Mehdi Universit éBadji Mokhtar-Annaba

Jury de Soutenance :

Mme Mohamed benali.Y	Pr	UBMA	Président
Mr Mendjel.M.S.M	MCB	UBMA	Encadrant
Mr Ghazi.S	MCB	UBMA	Examinateur

Année Universitaire : 2020/2021

RESUME

Le travail présenté dans ce manuscrit se situe dans le domaine des systèmes de recommandation qui est devenu une méthodologie dominante dans la majorité des applications Web, compris les sites web de commerce comme Amazon.com, les TV en ligne …etc., pour suggérer des articles et produits pour les consommateurs qui correspondent à leurs préférences et intérêts en expliquant la recommandation. Dans les recherches récentes, la performance des systèmes de recommandation a été limitée à leur capacité à prédire les items non vus par les utilisateurs et de leurs recommander les items avec les évaluations des prédictions les plus élevées. En plus de la performance de la prédiction, un bon recommandeur devrait offrir des items pertinents et divers qui correspondent aux différents choix et goûts d'intérêt des utilisateurs. La recommandation explicable tente de résoudre le problème du pourquoi, en fournissant des explications aux utilisateurs ou aux concepteurs de systèmes, elle aide les humains à comprendre pourquoi certains éléments sont recommandés par l'algorithme. Cependant, le manque de diversité et la stabilité des systèmes de recommandation par rapport au profil de l'utilisateur deviennent des défis majeurs. Nous proposons dans ce mémoire, deux contributions distinctes pour améliorer la diversité dans les listes de recommandation et rendre les recommandeurs plus adaptatifs aux changements des préférences des utilisateurs. Dans la première contribution, créer un système de recommandation à base de tags et la deuxième contribution consiste à intégrer la partie explication.

Mots clés. Système de Recommandation, Profil utilisateur, Filtrage Collaboratif, Filtrage basé Contenu, recommandation explication

ملخص

العمل المقدم في هذه مذكرة هو في مجال أنظمة التوصية التي أصبحت منهجية سائدة في غالبية تطبيقات الويب ، بما في ذلك مواقع التجارة مثل الأفلام والتلفزيون عبر الإنترنت ... إلخ ، لاقتراح المقالات والمنتجات للمستهلكين المتطابقة تفضيلاتهم واهتماماتهم مع شرح هذه التوصية. في الأبحاث الحديثة، اقتصر أداء أنظمة التوصية على قدرتها على التنبؤ بالعناصر التي لم يراها المستخدمون والتوصية بالعناصر لهم بأعلى درجات التنبؤ بالإضافة إلى أداء التنبؤ، يجب على المُوصي الجيد أن يقدم عناصر ذات صلة ومتنوعة تتوافق مع الخيارات والأذواق المختلفة التي تهم المستخدمين , تحاول التوصية القابلة للتفسير حل سبب المشكلة: من خلال تقديم تفسيرات للمستخدمين أو مصممي النظام، فإنها تساعد البشر على فهم سبب التوصية ببعض العناصر بواسطة الخوارزمية

ومع ذلك فإن الافتقار إلى التنوع واستقرار أنظمة التوصية فيما يتعلق بملف تعريف المستخدم أصبح من التحديات الرئيسية. في هذه الأطروحة ، نقترح مساهمتين متميزتين لتحسين التنوع في قوائم التوصيات وجعل المقترحين أكثر تكيفًا مع التغييرات في تفضيلات المستخدم. في المساهمة الأولى ، نظام التوصية. تتكون المساهمة الثانية شرح التوصية مع المستخدم.

الكلمات مرشدة : *نظام التوصيات, ملف تعريف المستخدم، التصفية التعاونية، التصفية القائمة على المحتوى، شرح التوصيات*

REMERCIEMENTS

En tout premier lieu, je remercie le bon Allah Tout Puissant, qui m'a donné la force, la volonté et le courage pour terminer ce modeste travail.

Je tiens à exprimer toute ma reconnaissance à mon directeur de mémoire, Dr. Mendjel Mehdi. Je le remercie de m'avoir encadré, orienté

, aidé et conseillé.

J'adresse mes sincères remerciements à tous les professeurs, intervenants et toutes les personnes qui par leurs paroles, leurs écrits, leurs conseils et leurs critiques ont guidé mes réflexions et ont accepté de me rencontrer et de répondre à mes questions durant mes recherches.

Je remercie mes très chers parents, qui ont toujours été là pour moi. Je remercie mes frères, pour leur encouragement.

Enfin, je remercie mes amies qui ont toujours été là pour moi. Leur soutien inconditionnel et leurs encouragements ont été d'une grande aide.

À tous ces intervenants, je présente mes remerciements, mon respect et ma gratitude.

DEDICACES

JE DÉDIE CE MÉMOIRE À :

AU MEILLEUR DES PÈRES

A MA TRÈS CHÈRE MAMAN

QU'ILS TROUVENT EN MOI LA SOURCE DE LEUR FIERTÉ

A QUI JE DOIS TOUT.

A MON FRÈRES ET MES SŒURS

SABAH, ABDHAK , HAROUN , RODWOIN

A QUI JE SOUHAITE UN AVENIR RADIEUX PLEIN DE RÉUSSITE

A MES AMIS

A TOUS CEUX QUI ME SONT CHERS.

TABLE DES MATIERES

Contenu

TABLE DES ILLUSTRATIONS

INTRODUCTION

Les systèmes de recommandation sont une forme spécifique de filtrage d'informations (SI) qui vise à présenter des éléments d'information (films, musique, livres, actualités, photos, pages Web, etc.) susceptibles d'intéresser l'utilisateur. Typiquement, un système de recommandation compare le profil d'un utilisateur à certaines caractéristiques standard et cherche à anticiper «l'opinion» que l'utilisateur pourrait fournir [01].

Les systèmes de recommandation sont nés de la volonté de pallier le problème de surcharge d'information du web et de certaines contraintes auxquelles se heurtent les outils existants de recherche d'informations. Combinant des techniques de filtrage d'information, personnalisation, intelligence artificielle, réseaux sociaux et interaction personne-machine, les systèmes de recommandation fournissent à des utilisateurs des suggestions qui répondent à leurs besoins et préférences informationnels. En effet, les systèmes de recommandation sont particulièrement sollicités dans les applications de commerce électronique. Par exemple, Amazon recommande toutes sortes de produits (films, musiques, livres, etc.), Netflix, Moviecritics, et Cinema. x suggèrent des films, TiV recommande des émissions de télévision, iTunes, Lastfm, Pandora et Rhapsody proposent de la musique, Trabble conseille des restaurants, Expedia et Travelocity recommandent des voyages, CNN suggère des nouvelles, Jester propose des blagues, Google recommande des pages web, WikiLens (www.wikilens.org) est une plateforme wiki expérimentale qui permet aux utilisateurs de recommander tout ce qu'ils désirent. Ils fournissent non seulement aux utilisateurs ou aux concepteurs de système des résultats de recommandation, mais également des explications pour clarifier pourquoi de tels éléments sont recommandés. La recommandation avec explication vise à répondre aux questions de type pourquoi dans les systèmes de recommandation.

Selon l'objectif précédemment cité, qui consiste à déterminer quelles ressources sont pertinentes pour un utilisateur, connaissant le contexte dans lequel il est immergé. Nos travaux sont axés vers cet objectif mais avec la possibilité de prédire la pertinence d'une ressource web pour un utilisateur particulier à partir de l'observation de son comportement et des éventuelles appréciations qu'il dépose lors de ses consultations, cela comprend également le développement de méthodes efficaces pour fournir les recommandations ou les explications

aux utilisateurs ou aux concepteurs de systèmes, car les recommandations explicables impliquent naturellement les humains dans la boucle.

La recherche montre que les explications aident les utilisateurs à prendre des décisions plus précises, à améliorer l'acceptation des recommandations par les utilisateurs et à accroître la confiance dans le système de recommandation.

Plusieurs raisons nous motivent à choisir ce sujet. Premièrement, nous aimerions étudier des algorithmes et des méthodes qui peuvent être appliquées dans le réel, surtout dans le développement de méthode efficaces dans le domaine de l'intelligence artificielle. Notre but principal est d'obtenir des expériences de recherche en informatique et des nouvelles connaissances dans ce domaine. Deuxièmement, l'objectif de nos recherches rentre dans le cadre de proposer un système de recommandation efficace avec explication qui rend l'utilisateur plus ouvert dans son choix.

Ce mémoire est organisé comme suit :

Le chapitre 1 : dans ce chapitre, nous présentons une vue générale sur les systèmes de recommandation ainsi que leurs avantages et inconvénients. Ensuite, nous exposons les métriques d'évaluation des systèmes de recommandation.

Le chapitre 2 : Le deuxième chapitre contient les méthodes utilisées pour la création des systèmes de recommandation en se focalisant principalement sur les deux méthodes célèbres qui ont prouvé leur efficacité dans le domaine des recommandations : la méthode du doc2vec et la méthode de Factorisation Matricielle (corrélation de Pearson). Ce chapitre s'achève par la représentation des tags et leur utilité dans le domaine de l'explication.

Le chapitre 3 : est consacré à la présentation et l'implémentation du système réalisé. Ensuite, nous discutons les résultats des expérimentations.

CHAPITRE 1
SYSTEMES DE RECOMMANDATION

1.1 Introduction

Ce chapitre présente les concepts de bases des systèmes de recommandation. Les méthodes de filtrages collaboratives, basées sur le contenu et hybrides seront décrites d'une façon plus détaillée. Finalement, le chapitre s'achève par les principales métriques et bases de données utilisées pour l'évaluation des systèmes de recommandation.

1.2 Définition des systèmes de recommandation

Les systèmes de recommandation peuvent être définis de plusieurs façons, vue la diversité des classifications proposées pour ces systèmes, mais il existe une définition générale de Robin Burke [02] qui les définit comme suit :"Des systèmes capable de fournir des recommandations personnalisées permettant de guider l'utilisateur vers des ressources intéressantes et utiles au sein d'un espace de données important". Les deux entités de base qui apparaissent dans tous les systèmes de recommandations sont l'usager et l'item. L'«usager» est la personne qui utilise un système de recommandation, donne son opinion sur divers items et reçoit les nouvelles recommandations du système. L'«Item» est le terme général utilisé pour désigner ce que le système recommande aux usagers.

Les données d'entrée pour un système de recommandation dépendent du type de l'algorithme de filtrage employé. Généralement, elles appartiennent à l'une des catégories suivantes :

- Les estimations :(également appelées les votes), expriment l'opinion des utilisateurs sur les articles (exemple : 1 mauvais à 5 excellent).
- Les données démographiques : se réfèrent à des informations telles que l'âge, le sexe, le pays et l'éducation des utilisateurs. Ce type de données est généralement difficile à obtenir et est normalement collecté explicitement.

- Les données de contenu : qui sont fondées sur une analyse textuelle des documents liés aux éléments évalués par l'utilisateur. Les caractéristiques extraites de cette analyse sont utilisées comme entrées dans l'algorithme de filtrage afin d'en déduire un profil d'utilisateur.

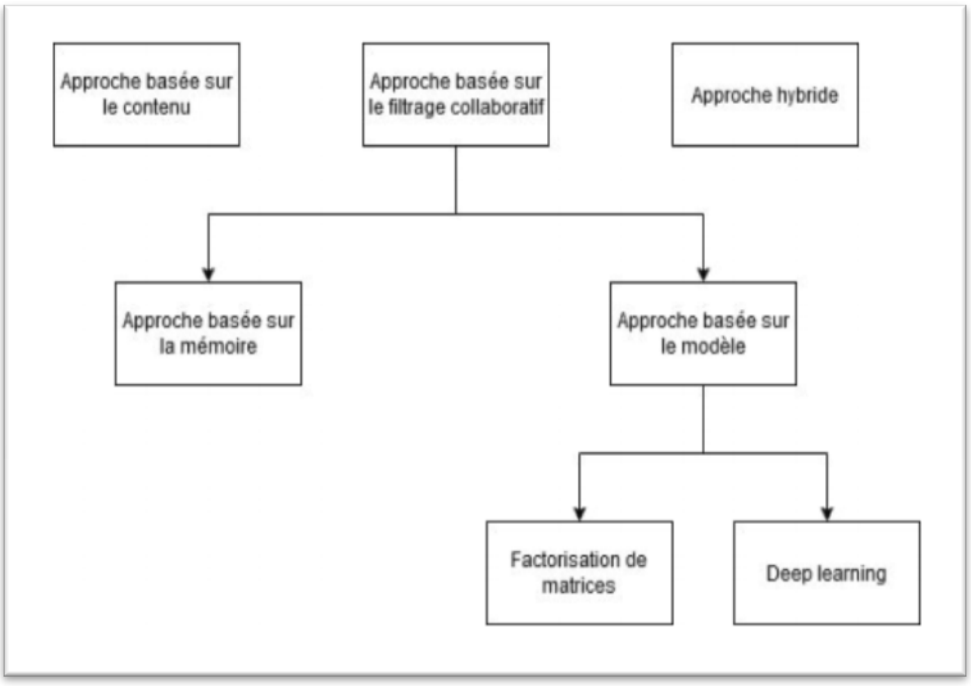

Figure 1: Les différentes approches des systèmes de recommandations [03]

1.3 Technique des systèmes recommandation

L'utilisation de techniques de recommandation efficaces et précises est très importante pour un système qui fournira des recommandations bonnes et utiles à ses utilisateurs individuels. Cela explique l'importance de comprendre les caractéristiques et les potentiels des différentes techniques de recommandation. Les figures 2 et 3 montrent l'anatomie des différentes techniques de filtrage de recommandation.

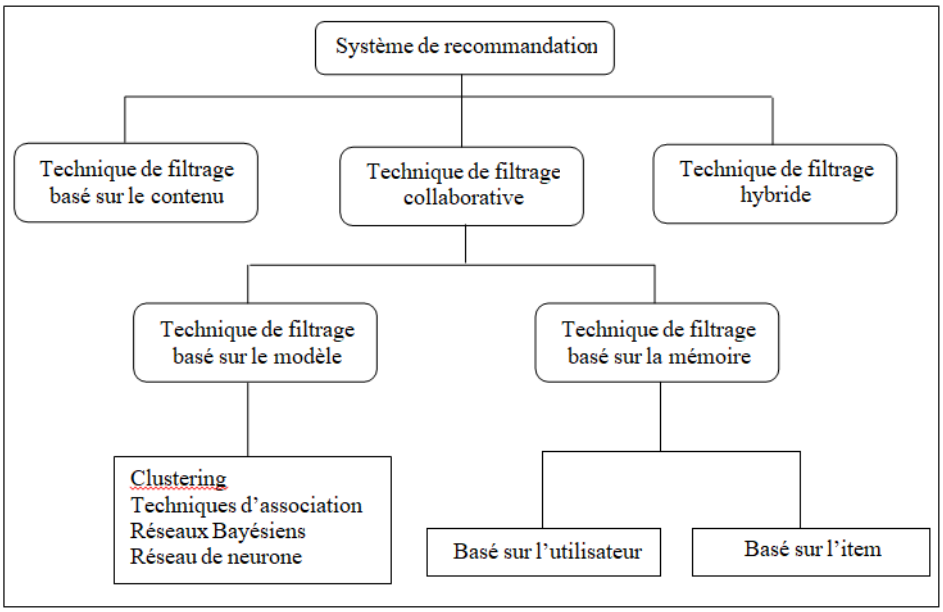

Figure 2 : Techniques de recommandation [04]

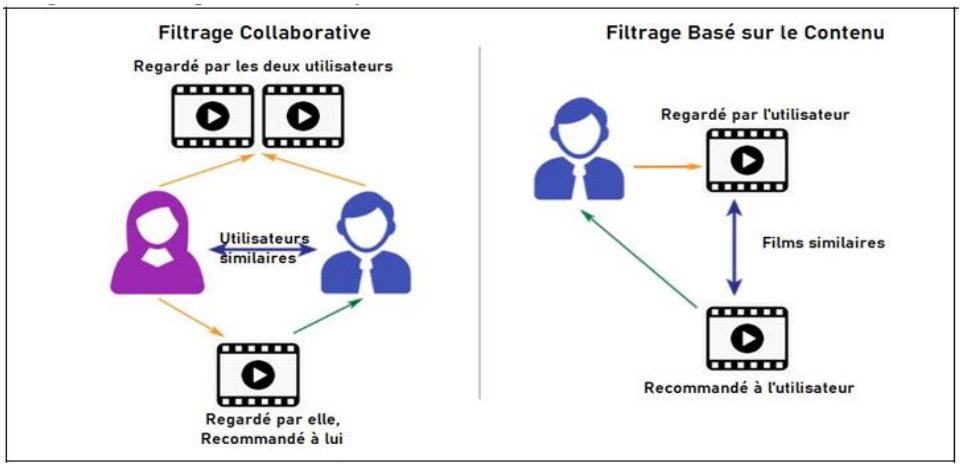

Figure 3 : Filtrage Collaborative / Filtrage Basé sur le Contenu [05]

1.3.1 Les Système de recommandation basé sur le contenu

Des recherches antérieures sur l'accès personnalisé à l'information s'étaient concentrées sur la combinaison des connaissances sur les éléments et des informations sur les préférences de l'utilisateur afin de localiser les éléments appropriés. Les premiers travaux cités dans [06] sur Grundy (recommandation des livres) et la méthode citée dans [07] (recherche d'informations) peuvent maintenant être considérés comme les premiers exemples des systèmes de recommandation bien que le terme n'ait pas encore été inventé. Cette approche, en raison de sa dépendance à la source de connaissances sur le contenu, en particulier aux fonctionnalités des éléments, est devenue la recommandation basée sur le contenu.

La recommandation basée sur le contenu est étroitement liée à l'apprentissage automatique supervisé. Nous pouvons voir le problème comme un problème d'apprentissage d'un

ensemble de classificateurs spécifiques à l'utilisateur où les classes sont « utiles à l'utilisateur X » et « non utiles à l'utilisateur X».

La correspondance entre l'ensemble de fonctionnalités et la fonction utilitaire de l'utilisateur doit également être bonne. Etant donné les informations sur le genre de film et sachant qu'un utilisateur aime « Spiderman » et « Black Panther», on peut en déduire une prédilection pour la science- fiction et donc recommander « Avengers : infinitywar». Les recommandeurs basés sur le contenu font référence à de telles approches, qui fournissent des recommandations en comparant les représentations de contenu décrivant un élément à des représentations de contenu qui intéressent l'utilisateur. Ces approches sont parfois également appelées filtrage par contenu [08].

Le CBF utilise différents types de modèles pour trouver des similitudes entre les items afin de générer des recommandations significatives. Il pourrait utiliser des modèles probabilistes tels que le classifier de Naïve Bayes, les arbres décisionnels ou les réseaux de neurone [09]pour modéliser la relation entre différents items au sein d'un corpus. Ces techniques font des recommandations en apprenant le modèle sous-jacent à l'aide de techniques d'analyse statistique ou d'apprentissage automatique.

Pour les recommandations basées sur le contenu, la tâche consiste à déterminer quels éléments du catalogue coïncident le mieux avec les préférences de l'utilisateur. Une telle approche ne requiert pas une grande communauté d'utilisateurs ou un gros historique d'utilisation du système.

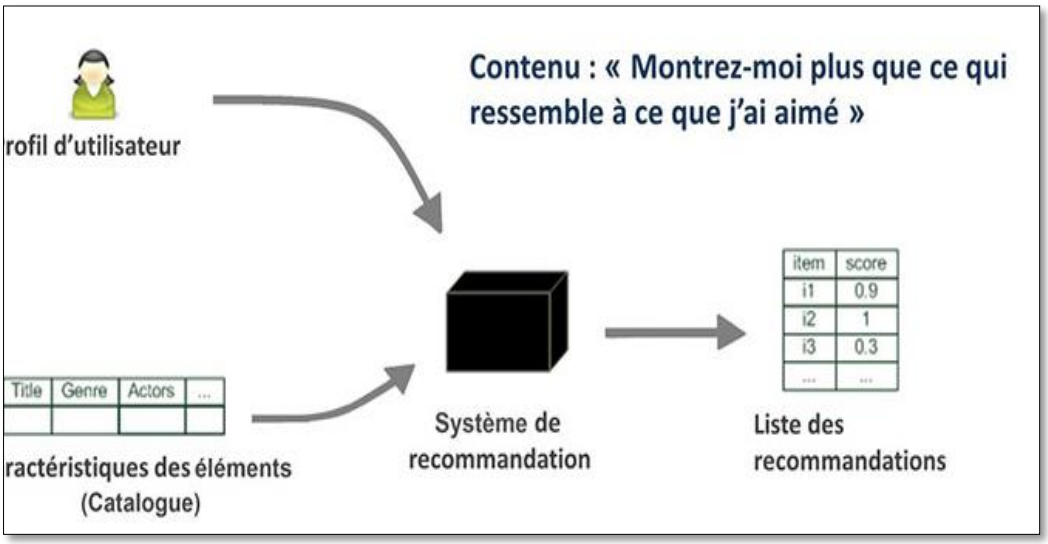

Figure 4 : Un système de recommandation basé sur le contenu [05]

Par exemple, on construit un profil d'un utilisateur « A » qui préfère les séries dont les genres préférés sont actions et romances (cas de Netflix). Et, on essaie de recommander des

produits qui de la même section que « A » préfère. (On ne se base plus sur les opinions des autres utilisateurs)[10].

1.3.1.1 Avantages et les inconvénients filtrage bases sur le contenu

Les avantages :

- Recommander des items similaires à ceux que les utilisateurs ont aimés dans le passé.
- Le matching entre les préférences de l'utilisateur et les caractéristiques des items fonctionne aussi pour les données textuelles.
- Pas besoin de données sur les autres utilisateurs.
- Possibilité de faire des recommandations à des utilisateurs avec des goûts « uniques ».
- Possibilité de recommander de nouveaux items ou même des items qui ne sont pas populaires [11].
- Possibilité de recommander des produits qui ne sont pas populaires ou nouveaux.

Les inconvénients :

- Tous les contenus ne peuvent pas être représentés avec des mots-clés (exemple: les images, …).
- Des items représentés par le même ensemble de mots-clés ne peuvent pas être distingués.
- Les utilisateurs avec des milliers d'achats/items sont un problème.
- Nouvel utilisateur : pas d'historique.
- "Over-specialization" : limitation aux items similaires.
- Les profils utilisateurs…
- Pour produire des recommandations précises, l'utilisateur doit fournir un « feedback » sur les suggestions retournées
- les utilisateurs n'aiment pas…

- Entièrement basé sur les scores d'articles et de sujets d'intérêt: moins il y a de scores, plus l'ensemble de recommandations possibles est limité.
- Il faut faire attention à l'évolution des intérêts de l'utilisateur et les prendre en compte.
- Impossible d'exploiter les jugements des autres utilisateurs.
- Trouver les bons "features" n'est pas toujours facile.
- Comment créer un profil pour les nouveaux utilisateurs [10].

1.3.2 Les Système de recommandation basés sur le filtrage collaboratif

Le filtrage collaboratif (de l'anglais : collaborative filtering) regroupe l'ensemble des méthodes qui visent à construire des systèmes de recommandation utilisant les opinions et évaluations d'un groupe pour aider l'individu [12]. Ce modèle s'appuie sur les appréciations données par un ensemble d'utilisateurs sur un ensemble d'articles. Ces appréciations, traduites en valeurs numériques peuvent être des notes, des comptes d'achats effectués, des nombres de visites, etc. On distingue deux grandes approches de filtrage collaboratif. L'approche se référant aux utilisateurs Resnick and Varian consiste à comparer les utilisateurs entre eux et à retrouver ceux ayant des goûts en commun, les notes d'un utilisateur étant ensuite prédites selon son voisinage. L'approche se référant aux articles Sarwar consiste à rapprocher les articles appréciés par les mêmes personnes et à prédire les notes des utilisateurs en fonction des articles les plus proches de ceux qu'ils ont déjà notés. Dans un système du filtrage collaboratif, il faut que les utilisateurs fournissent des évaluations des items qu'ils ont déjà utilisés, sous forme des notes, pour constituer leurs profils. Il n'y a aucune analyse du sujet ou du contenu des objets à recommander. Ce type de système est très efficace au cas où le contenu des objets est complexe, compliqué ou impossible à analyser, l'utilisateur peut apercevoir divers domaines intéressants, car le principe du filtrage collaboratif ne se fonde absolument pas sur la dimension thématique des profils, et n'est pas soumis à l'effet « entonnoir ».

Parmi les avantages du filtrage collaboratif les jugements de valeur des utilisateurs intègrent non seulement la dimension thématique mais aussi d'autres facteurs relatifs à la qualité des items tels que la diversité, la nouveauté, l'adéquation du public visé, etc. Un problème du système collaboratif est que sa performance dépend beaucoup de la distribution des évaluations (notes) données par utilisateurs. Dans le cas où il y a plusieurs items qui ont été

utilisés et évalués par très peu d'utilisateurs, ces items seraient recommandés très rarement, même si ces utilisateurs ont donné des notes très hautes pour ces items. Ce problème est connu comme le problème de parcimonie (sparsity problem). De la même façon, si dans le système il existe des utilisateurs qui ont des goûts très différents en comparaison avec les autres, le système ne peut pas trouver des similarités entre utilisateurs et donc ne peut pas donner des bonnes recommandations (voir figure 5).

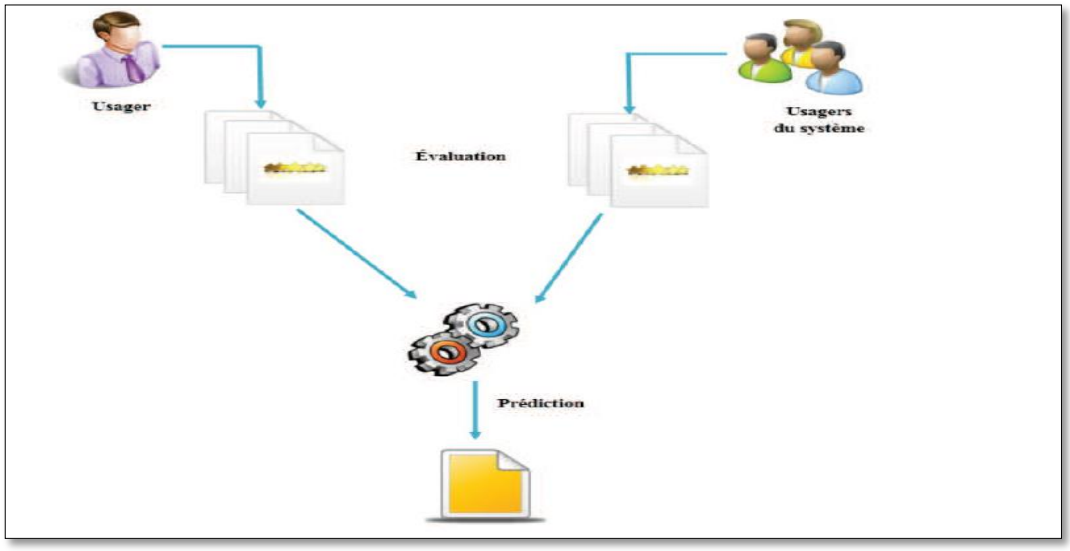

Figure 5: Recommandation basé sur le filtrage collaboratif [08]

La figure 6 représente un tableau de films avec sur un axe les utilisateurs d'un même Système (ex : un groupe d'amis sur MovieLens) et sur un autre les films. Chaque cellule de la matrice contient l'avis donné par un utilisateur pour un film, la cellule vide signifie qu'il n'a pas d'avis particulier sur le film. A fin de prédire si Illyes apprécierait le film "Harry Potter" et probablement lui recommander ce film, on compare les Votes d'Illyes à ceux des autres utilisateurs choisis. On peut alors voir qu'Illyes et Imen ont des Votes identiques,et que Imen n'a pas aimé le film «Harry Potter », on pourrait alors prédire qu'Illyes n'aimera pas aussi ce film et de ne lui pas faire cette suggestion.

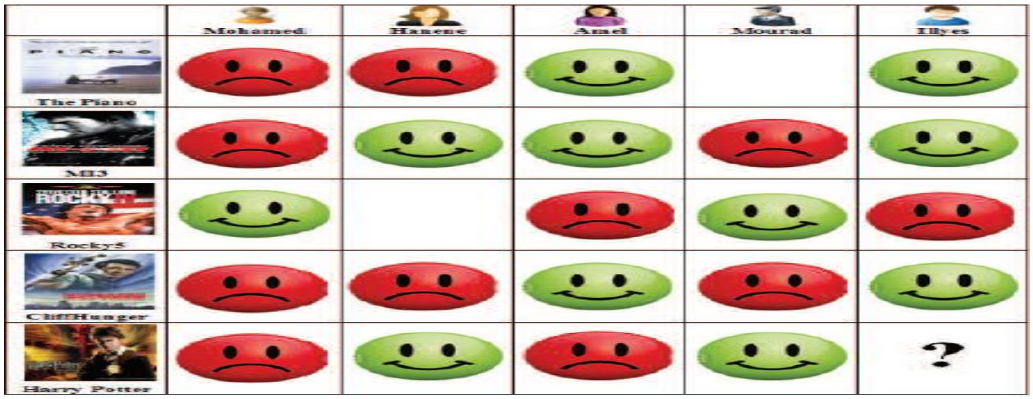

Figure 6 : Exemple de recommandation basé sur le filtrage collaboratif [08]

Le filtrage collaboratif est très utilisé par les systèmes vu ses avantages, parmi ces systèmes on peut citer : Amazon, Netflix, MovieLens, Jester, Citeseer, Ta-pestry, Phoaks, etc. L'exploitation des données disponibles dans un système de filtrage peut se faire de plusieurs manières. Ces méthodes sont classées en deux familles principales : les algorithmes basés mémoire est les algorithmes basés modèle [08].

1.3.2 .1 Le filtrage collaboratif base sur la mémoire

Le filtrage collaboratif basé sur la mémoire utilise une matrice des votes contenant des préférences des utilisateurs pour prédire des sujets additionnels ou des produits auxquels un nouvel utilisateur peut être intéressé. L'objectif d'un filtrage collaboratif basé sur la mémoire est de prédire l'utilité des ressources (items) pour un utilisateur particulier (l'utilisateur actif) basé sur la base des votes d'utilisateur. Ainsi, dans le filtrage basé sur la mémoire, les notes des utilisateurs stockées par le système sont directement utilisées pour prédire les notes pour de nouveaux items. Cela peut être fait de deux manières connues sous le terme de recommandations basées sur les utilisateurs ou recommandations basées sur les items [08].

1.3.2 .1.1 Le filtrage collaboratif base sur la mémoire (utilisateurs)

A	Item 1	Item 2	Item 3	Item 4	Item 5
User 1	9	3	?	7	10
User 2		2	6	7	9
User 3	9	3	9	7	
User 4	6	6		2	1
User 5		7	9	3	4

Figure 7 : construit une matrice : [Utilisateur x Produit] [08]

Le filtrage collaboratif basé sur l'utilisateur prédit l'intérêt d'un utilisateur pour un élément en fonction des informations de notation provenant de profils d'utilisateurs similaires.

- Il calcule la similitude entre les utilisateurs en comparant leurs notes sur le même élément
- Puis il calcule la note prévue pour un élément par l'utilisateur actif comme une moyenne pondérée des notes de l'élément par des utilisateurs similaires à l'utilisateur actif où les poids sont les similitudes de ces utilisateurs avec l'élément cible.

Soit « rx » le vecteur des notes de l'utilisateur x, et soit N l'ensemble des k utilisateurs les plus similaires à « x » qui ont noté l'élément « i », La prédiction de l'élément « i » pour l'utilisateur

« x » est calculée comme suit :

$$r_{xi} = \frac{1}{k} \sum_{y \in N} r_{yi} \qquad (1)$$

Oubien:

$$r_{xi} = \frac{\sum_{y \in N} s_{xy} \cdot r_{yi}}{\sum_{y \in N} s_{xy}} \qquad (2)$$

Où : Sxy = Sim(x,y)

Plusieurs types de mesures de similarité sont utilisés pour calculer la similarité entre les utilisateurs. Les deux mesures de similarité les plus populaires sont :

- **Mesure de similarité cosinus:**

Dans cette méthode les utilisateurs x et y sont considérés comme deux vecteurs de même origine dans un espace de « m » dimensions, est égale au nombre d'items évalués par les deux utilisateurs. Plus deux utilisateurs sont similaires, plus l'angle entre leurs vecteurs respectifs est petit. Empiriquement, la similarité entre ces deux utilisateurs est calculée par la formule du Cosinus suivante :

$$\mathbf{sim(x, y)} = \arccos(\boldsymbol{r_x}, \boldsymbol{r_y}) = \frac{r_x \cdot r_y}{||r_x|| \cdot ||r_y||} \qquad (3)$$

n : nombre d'items communs entre x et y votés par v.

rx: Vote de x pour l'item j.

ry: Vote de y pour l'item j.

- **Le coefficient de corrélation de Pearson:**

La corrélation de Pearson est une méthode issue des statistiques. Elle est aussi très utilisée dans le domaine des systèmes de recommandation pour mesurer la similarité entre deux utilisateurs. La formule suivante, nous donne cette valeur pour deux utilisateurs « x » et « y »:

$$\boldsymbol{sim(x, y)} = \frac{\sum_{s \in S_{xy}} (r_{xs} - \overline{r_x})(r_{ys} - \overline{r_y})}{\sqrt{\sum_{s \in S_{xy}} (r_{xs} - \overline{r_x})^2} \sqrt{\sum_{s \in S_{xy}} (r_{ys} - \overline{r_y})^2}} \qquad (4)$$

Où : Sxy = éléments notés par les deux utilisateurs x et y.

j : Nombre d'objets ayant été voté à la fois par x et y.

vx,j : Vote de A pour l'item j.

$v\bar{x}$,j : Moyenne des votes de x.

1.3.2 .1.2 Le filtrage collaboratif base sur la mémoire (Items)

La technique de filtrage basée sur les éléments calcule les prédictions en utilisant la similitude entre les éléments. Il construit un modèle de similitudes d'éléments en récupérant tous les éléments notés par un utilisateur actif à partir de la matrice d'éléments/ utilisateur, il détermine à quel point les éléments récupérés sont similaires à l'élément cible, puis il sélectionne les k éléments les plus similaires et leurs similitudes correspondantes sont également déterminées. La prédiction est faite en prenant une moyenne pondérée de la note des utilisateurs actifs sur les éléments similaires k. Il utilise les mêmes métriques de similarité et fonctions de prédiction que dans le modèle utilisateur-utilisateur [13].

$$r_{xi} = \frac{\sum_{j \in N(i;x)} s_{ij} \cdot r_{xj}}{\sum_{j \in N(i;x)} s_{ij}} \qquad \textbf{(5)}$$

Sij = similitude des éléments i et j

rxj = évaluation de l'utilisateur u sur l'élément j

N (i ; x) = les éléments notés par x similaires à i

1.3.2 .2 Le filtrage collaboratif base sur un modelé

Cette technique utilise les évaluations précédentes pour apprendre un modèle afin d'améliorer les performances de la technique de filtrage collaboratif. Le processus de création de modèle peut être réalisé à l'aide de techniques d'apprentissage automatique ou d'exploration de données. Ces techniques peuvent rapidement recommander un ensemble d'éléments pour le fait qu'elles utilisent un modèle pré-calculé et qu'elles se sont avérées produire des résultats de recommandation similaires aux techniques de recommandation basées sur les plus proches voisins. Des exemples de ces techniques incluent la technique de réduction de dimensionnalité telle que la décomposition en valeurs singulières (SVD), la régression et le clustering. Les techniques basées sur des modèles analysent la matrice d'éléments/utilisateurs pour identifier les relations entre les éléments ; ils utilisent ces relations pour comparer la liste des N premières recommandations. Les techniques basées sur des modèles résolvent les problèmes de parcimonie associés aux systèmes de recommandation [19].

1.3.2 .3 Avantages et inconvénients du filtrage collaboratif

Les méthodes de filtrage collaboratif présentent plusieurs avantages dont les plus importants sont :

- Effet de surprise (trouver autre chose que ce que l'on cherchait).
- Non nécessitée de la connaissance du domaine.
- Possibilité d'indexation de tout genre d'items.
- L'élimination du problème de la sur-spécialisation.
- La qualité des suggestions s'améliore avec le temps.

Cependant, l'utilisation des techniques de filtrage collaboratif peut entraîner plusieurs problèmes :

- Le démarrage à froid (un nouvel utilisateur qui n'a aimé aucun item ne peut pas recevoir de recommandation puisque le système ne connait pas ses goûts).
- La parcimonie (sparsity, le nombre d'items candidats à la recommandation est souvent énorme et les utilisateurs ne notent qu'un petit sous-ensemble des items disponibles).
- Le problème du mouton gris (gray sheep, n'auront pas beaucoup d'utilisateurs voisins. il sera donc difficile de faire des recommandations pertinentes pour ce genre d'utilisateurs).

1.3.3 Systèmes recommandation le filtrage hybride

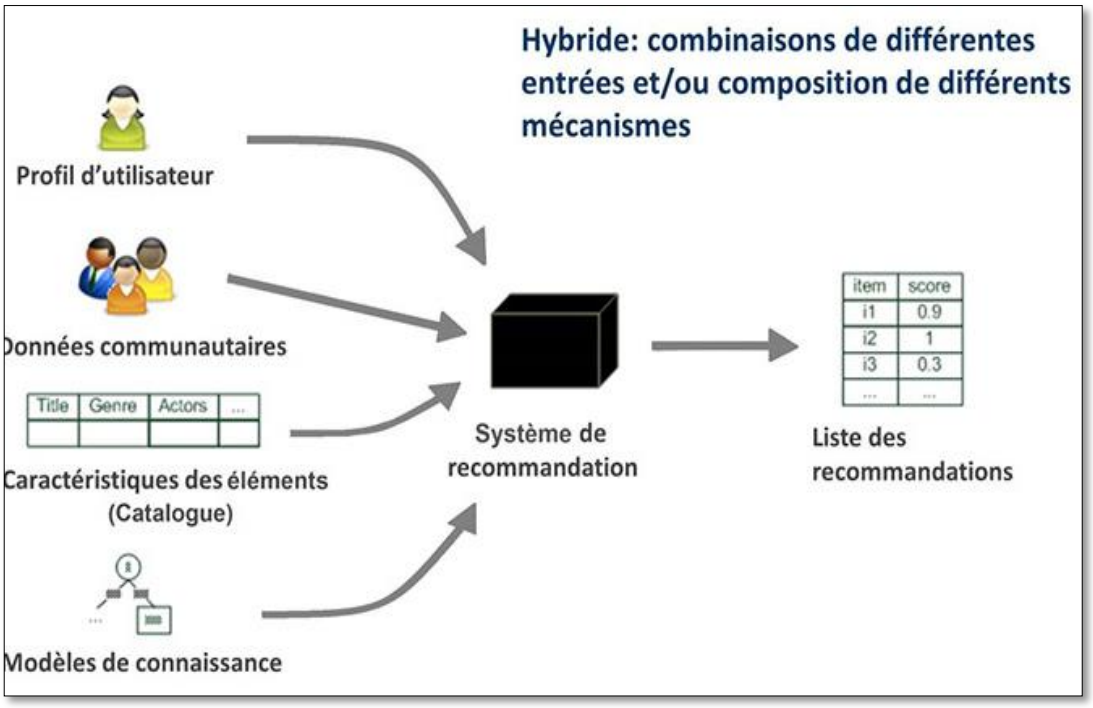

Figure 8 : Le système de recommandation hybride [05]

Constatant les avantages et les inconvénients de chacune des deux approche ci-dessus, on comprend que de nombreux systèmes reposent sur leur combinaison, ce qui en fait des systèmes de filtrage dits « hybrides ». En général, l'hybridation s'effectue en deux phases : (i) appliquer séparément le filtrage collaboratif et autres techniques de filtrage pour générer des recommandations candidates, et (ii) combiner ces ensembles de recommandations préliminaires selon certaines méthodes telles que la pondération, la mixtion, la cascade, la commutation, etc., afin de produire les recommandations finales pour les utilisateurs.

Plus généralement, les systèmes hybrides gèrent des profils d'utilisateurs orientés contenu, et la comparaison entre ces profils donne lieu à la formation de communautés d'utilisateurs permettant le filtrage collaboratif. La meilleure description des méthodes hybrides a été faite par Burke [02].Alors, selon Burke on peut distinguer sept façons de combiner les méthodes traditionnelles :

- **Pondération (Weighted)**

Une méthode hybride qui combine la sortie d'approches distinctes, utilisant, par exemple, une combinaison linéaire des scores de chaque technique de recommandation.

- **Commutation (Switching)**

C'est une technique qui permet de faire le choix d'un modèle de recommandation parmi plusieurs, en se basant sur plusieurs critères. La détermination de la technique appropriée dépend de la situation. Le système se doit alors de définir les critères de commutation, ou les cas où l'utilisation d'une autre technique est recommandée. Ceci permet au système de connaitre les points forts et les points faibles des techniques de recommandation qui le constituent.

- **Technique mixte (Mixed)**

Dans cette approche, le recommandeur ne combine pas, mais augmente la description des ensembles de données, en prenant en considération les estimations des utilisateurs et la description des items. La nouvelle fonction de prédiction doit faire face aux deux types de descriptions et permet d'éviter les problèmes posés par le filtrage collaboratif, à savoir, le démarrage à froid.

- **Combinaison de caractéristiques (Features combination)**

Dans un hybride basé sur la combinaison de caractéristiques, les données provenant de techniques collaboratives sont traitées comme une caractéristique, et une approche basée sur le contenu est utilisé sur ces données.

- **Cascade**

La cascade implique un processus étape par étape. Dans ce cas, une technique de recommandation est appliquée en premier, produisant un ensemble de candidats potentiels. Puis, une deuxième technique raffine les résultats obtenus dans la première étape. Cette méthode a pour avantage que si la première technique génère peu de recommandations, ou si ces recommandations sont ordonnées afin de permettre une sélection rapide, la deuxième technique ne sera plus utilisée.

- **Augmentation de caractéristiques (Feature augmentation)**

L'augmentation de caractéristiques est semblable à la cascade, mais dans ce cas-là les résultats obtenus (le classement ou la classification) de la première technique sont utilisées par le deuxième comme une caractéristique ajoutée.

- **Méta niveau (Meta-level)**

Dans un hybride basé sur méta niveau, une première technique est utilisée, mais différemment que la précédente méthode (augmentation de caractéristiques), non pas pour produire de nouvelles caractéristiques, mais pour produire un modèle. Dans la deuxième étape, c'est le modèle entier qui servira d'entrée pour la deuxième technique.

1.4 Métriques d'évaluation du système recommandation

La qualité d'un algorithme de recommandation peut être évaluée à l'aide de différents types de mesure qui peuvent être la précision ou la couverture. Le type de métrique utilisé dépend du type de technique de filtrage. La précision est la fraction de recommandations correctes sur le total des recommandations possibles, tandis que la couverture mesure la fraction d'objets dans l'espace de recherche pour lesquels le système est en mesure de fournir des recommandations. Les métriques pour mesurer la précision des systèmes de filtrage des recommandations sont divisées en métriques de précision statistiques et d'aide à la décision [25]. L'adéquation de chaque métrique dépend des caractéristiques de l'ensemble de données et du type de tâches que le système de recommandation effectuera.

- **Les mesures de précision statistique**

Les mesures de précision statistique évaluent la précision d'une technique de filtrage en comparant les évaluations prévues directement avec l'évaluation réelle de l'utilisateur. L'erreur absolue moyenne (MAE) [14], l'erreur quadratique moyenne (RMSE) et la corrélation sont généralement utilisées comme mesures de précision statistique. MAE est le plus populaire et le plus couramment utilisé ; il s'agit d'une mesure de l'écart de la recommandation par rapport à la valeur spécifique de l'utilisateur. Il est calculé comme suit [15]:

$$\mathrm{MAE} = \frac{1}{N}\sum_{u,i}|p_{u,i} - r_{u,i}| \qquad \textbf{(7)}$$

où $P_{u,i}$ est la note prédite pour l'utilisateur « u » sur l'élément « i »,
$r_{u,i}$ est la note réelle et N est le nombre total de notes sur l'ensemble d'articles. Plus le MAE est bas, plus le moteur de recommandation prédit avec précision les évaluations des utilisateurs.
En outre, l'erreur quadratique moyenne (RMSE) est donnée par Cotter et al [16] comme

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{u,i}(p_{u,i} - r_{u,i})^2} \qquad \textbf{(8)}$$

L'erreur quadratique moyenne (RMSE) met davantage l'accent sur une erreur absolue plus grande, et plus le RMSE est bas, meilleure est la précision de la recommandation.

- **Les mesures de précision et d'aide à la décision**

Les mesures de précision d'aide à la décision couramment utilisées sont le taux d'inversion, les erreurs pondérées, les caractéristiques de fonctionnement du récepteur (ROC) et la courbe de rappel de précision (PRC), la précision, le rappel et la mesure F. Ces métriques aident les utilisateurs à sélectionner des éléments de très haute qualité parmi l'ensemble d'éléments disponibles [16]. Les métriques considèrent la procédure de prédiction comme une opération binaire qui distingue les bons éléments de ceux qui ne le sont pas. Les courbes ROC sont très efficaces lors de l'exécution d'évaluations complètes des performances de certains algorithmes spécifiques. La précision est la fraction des éléments recommandés qui est réellement pertinente pour l'utilisateur, tandis que le rappel peut être défini comme la fraction des éléments pertinents qui font également partie de l'ensemble des éléments recommandés [17]. Ils sont calculés comme suit:

$$\mathrm{Precision} = \frac{\text{Correctly recommended items}}{\text{Total recommended items}}$$

$$\mathrm{Recall} = \frac{\text{Correctly recommended items}}{\text{Total useful recommended items}} \qquad \textbf{(9), (10)}$$

La mesure F définie ci-dessous permet de simplifier la précision et le rappel en une seule métrique. La valeur résultante rend la comparaison entre les algorithmes et entre les ensembles de données très simple et directe.

$$F\text{-measure} = \frac{2\,PR}{P+R} \qquad \textbf{(11)}$$

- **La couverture**

La couverture à avoir avec le pourcentage d'articles et d'utilisateurs qu'un système de recommandation peut fournir des prédictions. La prédiction peut être pratiquement impossible à faire si aucun utilisateur ou peu d'utilisateurs ont évalué un élément. La couverture peut être réduite en définissant de petites tailles de quartier pour l'utilisateur ou les éléments [18].

1.4Conclusion

Dans ce chapitre, nous avons présenté le problème de la recommandation. Les principaux composants d'un système de recommandation qui sont les utilisateurs et les items. Plus précisément, nous avons passé en revue les classes de systèmes de recommandations : les systèmes basés sur le filtrage collaboratif, les systèmes basés sur le filtrage à base de contenu et les systèmes hybrides.

CHAPITRE 2
LES METHODES ET EXPLICATION

2.1Introduction

Dans ce chapitre, en utilise la méthode qui faire dans ce système puis en présentera en détail la partie explication des systèmes de recommandation.

2.2 La factorisation matricielle

Une matrice d'évaluation est formée par une matrice R, dans laquelle les entrées correspondent aux notes de l'utilisateur pour un élément. Pour de nombreuses applications Internet, ces matrices sont volumineuses, avec des millions d'utilisateurs et d'éléments différents. Elles sont également creuses, ce qui signifie que chaque utilisateur n'a évalué, consulté ou acheté qu'un petit nombre d'éléments par rapport à l'ensemble. La grande majorité des entrées de la matrice, souvent plus de 99 % d'entre elles, présentent la valeur 0 [21].

La méthode de factorisation matricielle suppose qu'il existe un ensemble d'attributs communs à tous les éléments, chacun d'eux variant en fonction du degré avec lequel ils expriment ces attributs. De plus, la méthode suppose que chaque utilisateur a sa propre expression pour chacun de ces attributs, indépendamment des éléments. De cette manière, on peut estimer la note donnée par l'utilisateur à l'élément en faisant la somme de la force de chaque attribut utilisateur pondérée par le degré d'expressivité de cet attribut. Ces attributs sont parfois appelés facteurs masqués ou latents [21].

Intuitivement, il est facile de voir que ces facteurs latents hypothétiques existent réellement. Dans le cas des films, il est clair que de nombreux utilisateurs préfèrent certains genres, acteurs ou réalisateurs. Ces catégories représentent des facteurs latents qui, bien qu'évidents, sont toujours très utiles. Par exemple, les préférences de genre se manifestent dans les films que les utilisateurs ont tendance à aimer. Les personnes qui apprécient le même genre aiment vraisemblablement des films similaires. La puissance de la factorisation matricielle pour le filtrage collaboratif provient surtout du fait qu'il n'est pas nécessaire de connaître le nombre de

genres, d'acteurs ou d'autres catégories pouvant constituer la totalité des préférences d'un utilisateur donné. On suppose simplement qu'il y en a un nombre arbitraire [20].

La factorisation matricielle est une technique permettant de découvrir les facteurs latents à partir de la matrice de notation et de faire correspondre les éléments et les utilisateurs à ces facteurs. Considérons une matrice de notation R avec des notes par n utilisateurs pour m éléments. La matrice de notation R aura n × m lignes et colonnes. La matrice R peut être décomposée en deux matrices minces P et Q. P aura n × f dimensions et Q aura m × f dimensions où f est le nombre de facteurs latents. Dans l'exemple utilisé sur la figure 6, il existe deux facteurs latents. La matrice R peut être décomposée de telle manière que le produit scalaire de la matrice P et Q transposé donne une matrice de dimensions n × m qui se rapproche étroitement de la matrice de notation originale R tel que [21]:

$$\widehat{r_{ui}} = p_u \cdot q_i^T \quad \textbf{(12)}$$

L'objectif de la méthode de factorisation matricielle est d'apprendre les vecteurs P et Q à partir de la matrice de notation R, où P exprime la notation d'un élément en termes de facteurs f et Q exprime l'intérêt des utilisateurs pour les facteurs P et Q doivent être appris de telle manière que l'on puisse minimiser le delta entre les notations connues et les notations prévues. Supposons que K est l'ensemble des cellules connues et non vides dans la matrice de notation R. L'objectif est de minimiser l'erreur de prédiction ou la fonction de perte,

$$\min \sum_{(u,i) \in K} (r_{ui} - \widehat{r_{ui}})^2$$
$$\min \sum_{(u,i) \in K} (r_{ui} - p_u q_i^T)^2 \quad \textbf{(13)}$$

Où K est l'ensemble des (u, i) paires de notations connues.

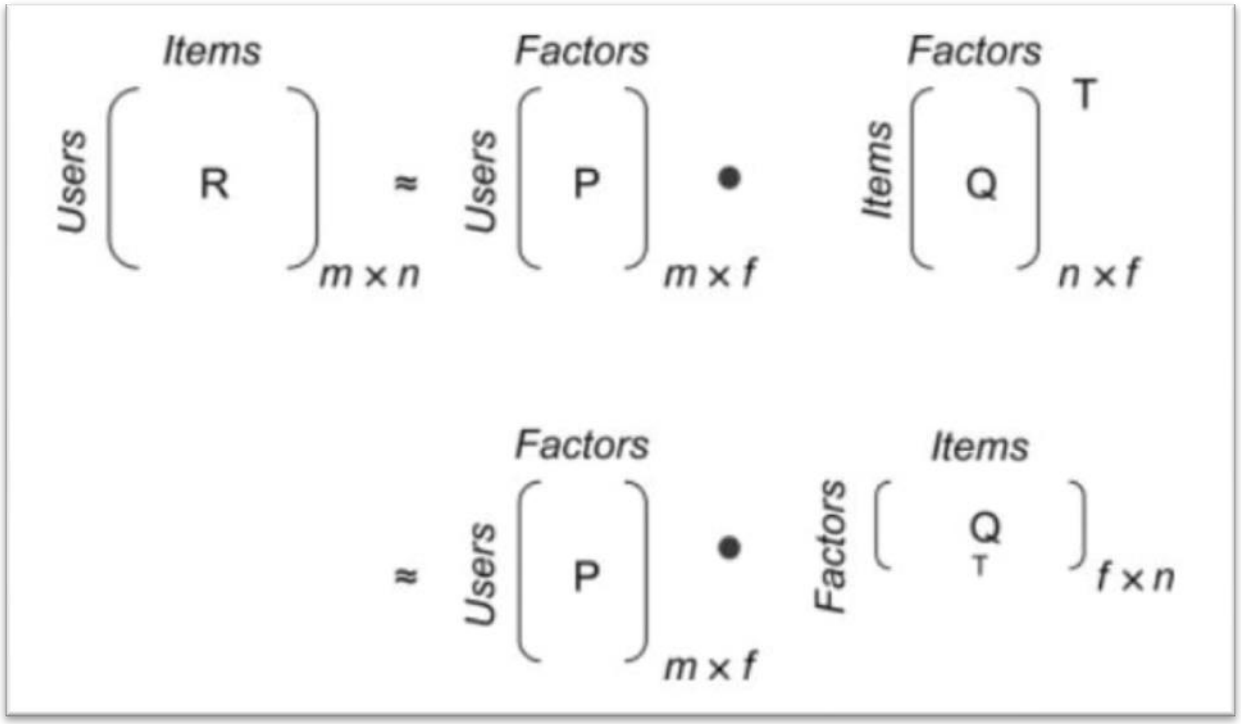

Figure 9 : Décomposition de la matrice de notation en matrices de facteurs latents. [22]

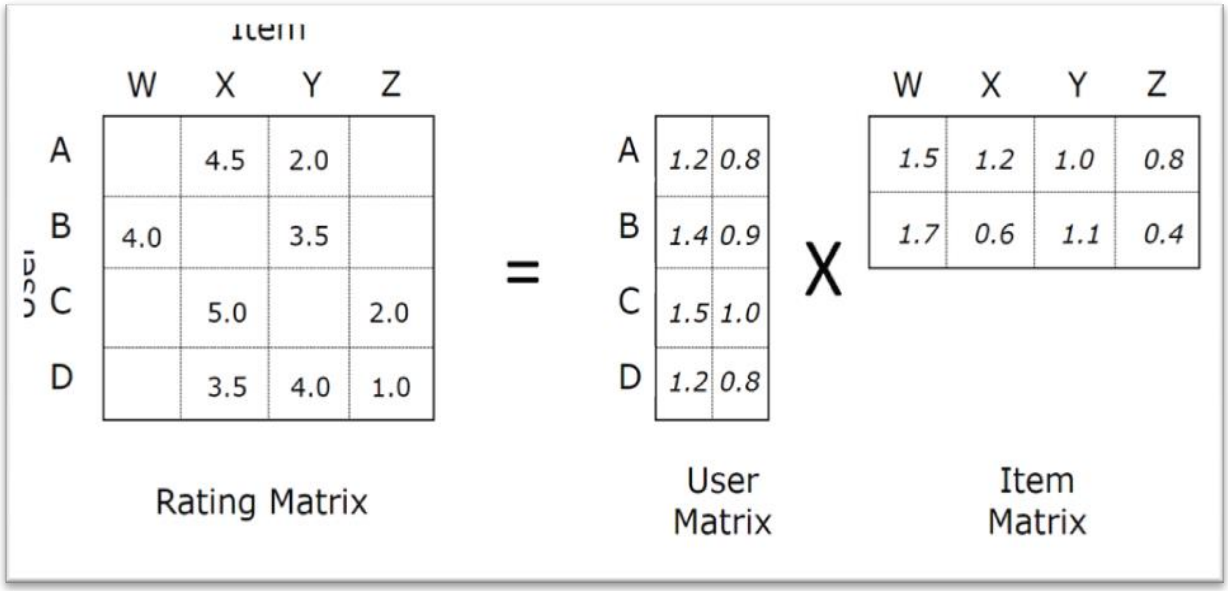

Figure 10 : matrix factorisation [23]

2.3 Doc2Vec

La motivation initiale derrière la construction de doc2vec était la nature non structurée des documents par rapport aux mots individuels. Doc2vec a été créé par Mikilov et Le en 2014 [26]. Mikolov était également l'un des auteurs de la recherche word2vec originale, qui est un autre indicateur que doc2vec s'appuie sur l'architecture word2vec.

Doc2vec est un outil NLP pour représenter des documents sous forme de vecteur et est une

généralisation de la méthode word2vec [27].

Il existe deux architectures principales pour la mise en œuvre de doc2vec : à savoir le modèle de mémoire distribuée des vecteurs de paragraphe (PV-DM) et la version de sac de mots distribués du vecteur de paragraphe (PV-DBOW).

- **Word2Vec**

C'est un modèle pour créer les incorporations de mots, où il prend l'entrée comme un grand corpus de texte et produit un espace vectoriel typiquement de plusieurs centaines de dimensions. Il a été présenté dans deux articles entre septembre et octobre 2013, par une équipe de chercheurs de Google. L'hypothèse sous-jacente de Word2Vec est que deux mots partageant des contextes similaires partagent également une signification similaire et par conséquent une représentation vectorielle similaire du modèle.

Par exemple : « Banque », « argent » et « comptes » sont souvent utilisés dans des situations similaires, avec des mots environnants similaires comme « dollar », « prêt » ou « crédit », et selon Word2Vec, ils partageront donc une représentation vectorielle similaire. À partir de cette hypothèse, Word2Vec peut être utilisé pour découvrir les relations entre les mots d'un ensemble de données, calculer la similitude entre eux ou utiliser la représentation vectorielle de ces mots comme entrée pour d'autres applications telles que la classification de texte ou le regroupement [28].

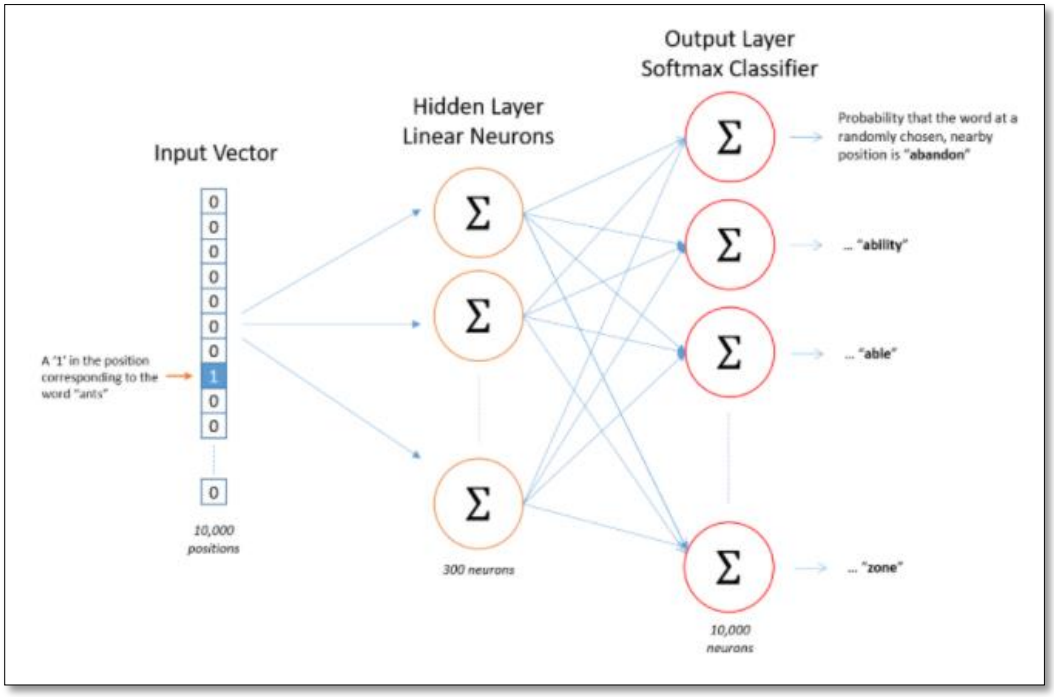

Figure 11 : Word2Vec [28]

- **Distributed Memory Version of Paragraph Vector (PV-DM)**

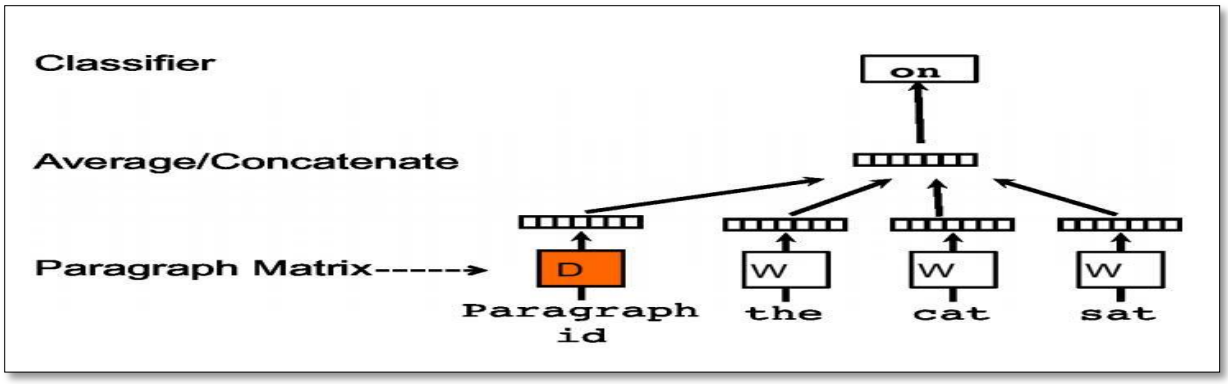

Figure 12 : PV-DM [28]

- Distributed Bag of Words Version of Paragraph Vector (PV-DBOW)

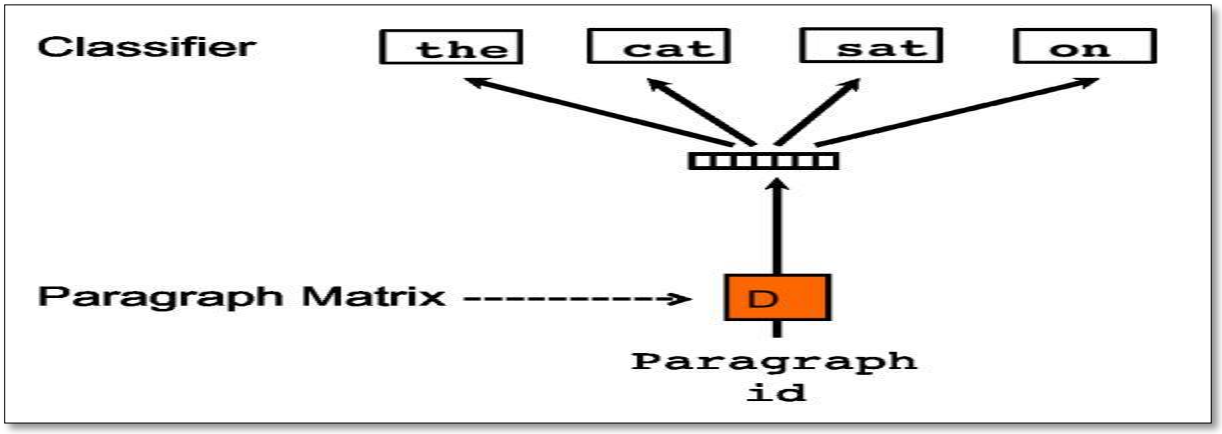

Figure 13 : PV-DBOW [28]

2.4 Explication des recommandations des SRs

Alors qu'une grande partie de la recherche sur les systèmes de recommandation (SRs) s'est concentrée sur l'amélioration de l'exactitude des recommandations, des travaux récents comme dans [29] suggèrent un ensemble plus large d'objectifs comprenant la confiance, la satisfaction des utilisateurs et la transparence. Une clé pour atteindre cet ensemble plus large d'objectifs est d'expliquer les recommandations aux utilisateurs. Alors que les recommandations indiquent aux utilisateurs quels éléments ils pourraient aimer, les explications révèlent pourquoi ils pourraient les aimer. Un exemple est la fonctionnalité «Pourquoi cela a été recommandé ?». On peut prendre comme exemple la plate-forme*Netflix* qui explique les recommandations de films en montrant aux utilisateurs des films similaires qu'ils ont bien notés dans le passé.

Il y a aussi un nouveau type d'explication qui utilise les tags comme fonctionnalités, appelé « tag plantation ». L'entité intermédiaire pour tags plantations est un tag ou un ensemble de tag, par exemple: "Nous recommandons le film Fargo car il est tagué avec quirky et vous avez

apprécié d'autres films tagués avec quirky".
Dans la suite de notre travail, nous étudions l'aspect des explications basées sur les tags et les notes cela en exploitant la pertinence du tag. La pertinence des tags représente le degré auquel un tag décrit un élément donné. Par exemple, considérons un tag pour le film « Pulp Fiction » qui utilise la balise «*darkcomedy*». Dans cet exemple, la pertinence des balises mesurerait à quel point la «*darkcomedy*» décrit Pulp Fiction.

2.4.1 Définition la recommandation explicable

La recommandation explicable fait référence à la recommandation personnalisée, ce sont des algorithmes qui répondent au problème du pourquoi - ils fournissent non seulement aux utilisateurs ou des concepteurs de systèmes avec des résultats de recommandations, mais aussi des explications pour clarifier pourquoi de tels articles sont recommandés. De cette façon, ils aident à améliorer la transparence, la persuasion, l'efficacité, la fiabilité, et la satisfaction des utilisateurs des systèmes de recommandation [30].

2.4.2 Information source for explication

Une explication est un élément d'information affiché aux utilisateurs, expliquant pourquoi un élément particulier est recommandé [30]. Des explications de recommandation peuvent être générées à partir de différentes sources d'informations et être présentées dans différents styles d'affichage, par exemple un utilisateur ou un élément pertinent, un graphique radar, une phrase, une image ou un ensemble de règles de raisonnement. En outre, il pourrait exister de nombreuses explications différentes pour une même recommandation

2.4.4 Relation entre l'item et tag : relevance tag

Nous utilisons le terme relevance tag cité dans [31] pour décrire la relation entre un tag et un élément. Une mesure possible de la relevance est la popularité des tags. Autrement dit, de nombreux utilisateurs ont-ils appliqué un tag donné à un élément ou seulement quelques utilisateurs? Un tag appliqué par de nombreux utilisateurs est probablement plus pertinent pour l'élément donné.

2.5 Conclusion

Dans ce chapitre, nous avons exposé et expliqué les différentes phases pour la réalisation de notre système de recommandation, qui permet de recommander une liste pertinente de documents à un utilisateur donné puis expliquer les recommandations suggérées.

CHAPITRE 3
IMPLEMENTATION ET EVALUATION

3.1 Introduction

Ce chapitre présenté la mise en place de notre système de recommandation avec explication, la première étape de cette mise en œuvre était l'exploration des données, pour se faire on a utilisé le data-set movielens avec les tags. Dans ce qui suit, on va présenter en détail l'approche utilisée puis la phase de l'évaluation.

3.2 Langage de programmation Python

Dans la partie implémentation nous avons utilisé le langage « Python » qui est un langage de programmation interprété, multi-paradigme et multiplateformes. Il favorise la programmation impérative structurée, fonctionnelle et orientée objet. Il est doté d'un typage dynamique fort, d'une gestion automatique de la mémoire par ramasse-miettes et d'un système de gestion d'exceptions ; il est ainsi similaire à Perl, Ruby, Scheme, Smalltalk et Tcl [08].

Python est un langage de programmation qui peut s'utiliser dans de nombreux contextes et s'adapter à tout type d'utilisation grâce à des bibliothèques spécialisées. Il est cependant particulièrement utilisé comme langage de script pour automatiser des tâches simples mais fastidieuses, comme un script qui récupérerait la météo sur Internet ou qui s'intégrerait dans un logiciel de conception assistée par ordinateur afin d'automatiser certains enchaînements d'actions répétitives (voir la section Adoption). On l'utilise également comme langage de développement de prototype lorsqu'on a besoin d'une application fonctionnelle avant de l'optimiser avec un langage de plus bas niveau. Il est particulièrement répandu dans le monde scientifique, et possède de nombreuses bibliothèques optimisées destinées au calcul numérique [32].

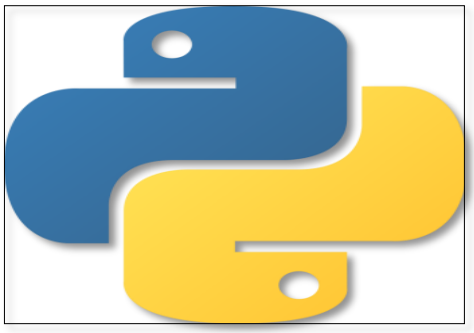

Figure 14 : Python Depuis : [33]

3.3 Environnement de développement

Nous présentons ci-dessous, les outils que nous avons utilisés pour créer notre système de recommandation.

3.3.1 Jupyter Notebook

Jupyter est une application web utilisée pour programmer dans plus de 40 langages de programmation, dont Python, Julia, Ruby, R, ou encore Scala2. C'est un projet communautaire dont l'objectif est de développer des logiciels libres, des formats ouverts et des services pour l'informatique interactive. Jupyter est une évolution du projet IPython. Jupyter permet de réaliser des calepins ou notebooks, c'est-à-dire des programmes contenant à la fois du texte en markdown et du code en Julia, Python, R... Ces calepins sont utilisés en science des données pour explorer et analyser des données [34].

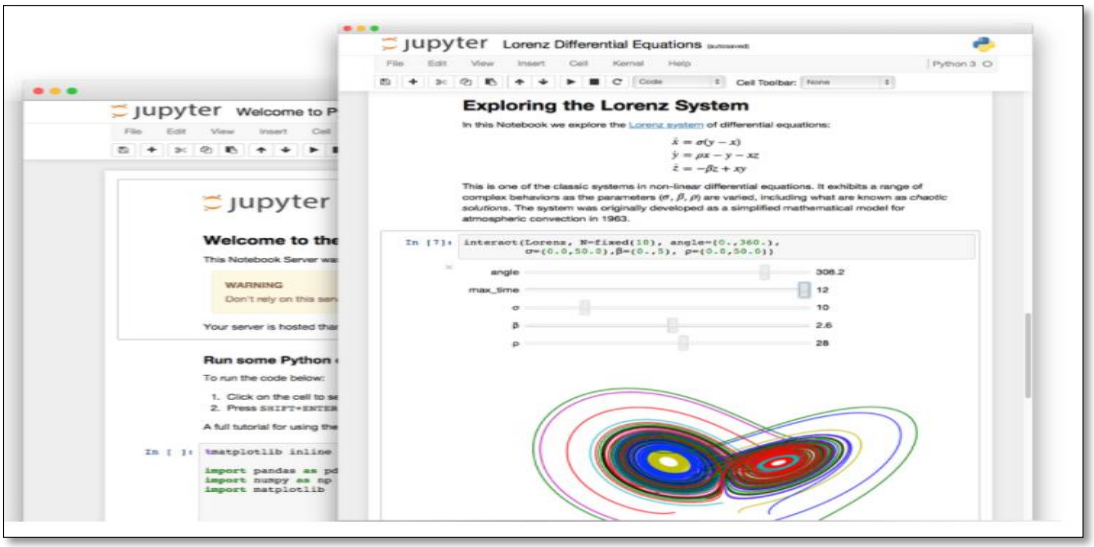

Figure 15 : platform Jupter notebooks [35]

3.3.2 Anaconda Navigator

Anaconda est une distribution des langages de programmation Python et R pour l'informatique scientifique (science des données, applications d'apprentissage automatique, traitement de données à grande échelle, analyse prédictive, etc.), qui vise à simplifier la gestion et le déploiement des packages.

La distribution comprend des packages scientifiques de données adaptés à Windows, Linux et MacOS. Il est développé et maintenu par Anaconda, Inc., fondé par Peter Wang et Travis Oliphant en 2012. En tant que produit Anaconda, Inc., il est également connu sous le nom de Distribution Anaconda ou d'une édition individuelle Anaconda, tandis que d'autres produits de la société sont Anaconda Team Edition et Anaconda Enterprise Edition, qui ne sont pas gratuites.

Les versions de forfait dans Anaconda sont gérées par le système de gestion de paquets Conda. [9] Ce gestionnaire de paquets a été renvoyé sous forme de package open-source séparé car il a fini par être utile seul et pour d'autres choses que Python. Il y a aussi une petite version bootstrap d'Anaconda appelée Miniconda, qui ne comprend que Conda, Python, les forfaits dont ils dépendent et un petit nombre d'autres paquets [36].

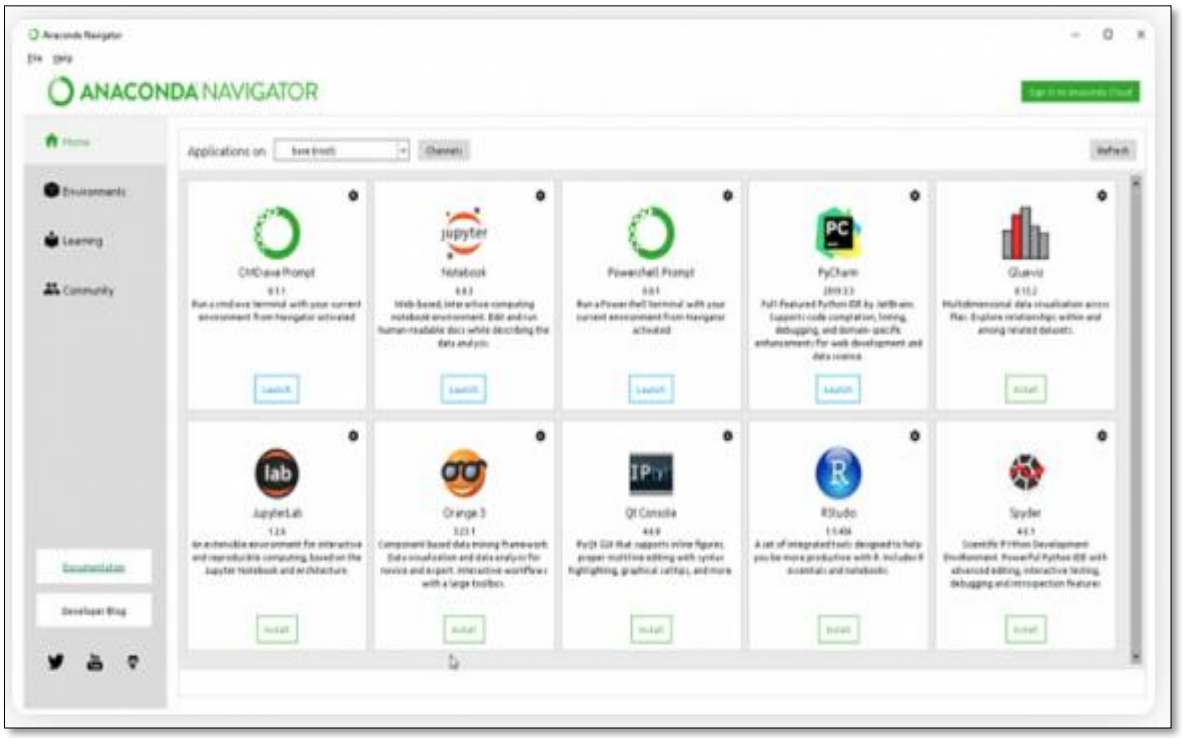

Figure 16 : interfaces anacondas Navigator [37]

3.4 Processus de mise en œuvre du système de recommandation

Pour la création d'un système de recommandation, il est nécessaire de suivre étape par étape une méthodologie bien précise depuis l'acquisition des données jusqu'à la recommandation finale et en fin l'explication de cette recommandation.

- Exploration des données.
- La sélection des films similaires par les tags.
- La sélection des films similaires par les évaluations.
- Intersection entre les deux méthodes.
- Explication de la recommandation.
- Interprétation des résultats.

3.4.1 Exploration des données

Maintenant que nous avons notre jeu de données, nous allons explorer ce qu'il contient. Cette analyse va nous permettre de comprendre la structure de nos données et de tirer des statistiques qui pourront nous servir à l'interprétation des résultats. Notre jeu de données contient des fichiers textuels ; movies.csv ratings.csv genome-scores.csv genome-tags.csv.

Cet ensemble de données (ml-20m) décrit l'activité de notation et de marquage de texte libre de MovieLens (entre 5 étoiles et 0.5 étoile). Il contient 25000095 rates et 1093360 applications des tags sur 62423 films. Ces données ont été créées par 162541 utilisateurs entre le 09 janvier 1995 et le 21 novembre 2019. Cet ensemble de données a été généré le 21 novembre 2019.

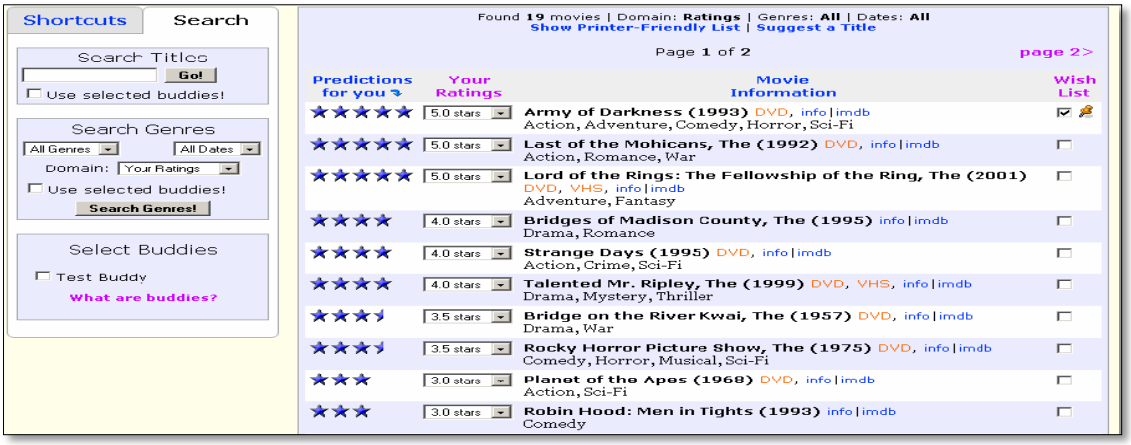

Figure 17 : group Movie Lens

- **Le fichier genome-scores.csv**

Le fichier genome-scores.csv nous donne la pertinence des tags (relevance) pour chaque film, cette pertinence est donnée entre (0-1) (voir figure 20).

```
In [11]: # movie tags
         genome_scores = pd.read_csv('C:/Users/info/Python/Code_Source_de_Memoire/ml-20m/genome-scores.csv')
         genome_scores.head(20)
```

Out[11]:

	movieId	tagId	relevance
0	1	1	0.02500
1	1	2	0.02500
2	1	3	0.05775
3	1	4	0.09675
4	1	5	0.14675
5	1	6	0.21700
6	1	7	0.06700
7	1	8	0.26275
8	1	9	0.26200
9	1	10	0.03200
10	1	11	0.57700
11	1	12	0.11625
12	1	13	0.18800
13	1	14	0.00800
14	1	15	0.03675
15	1	16	0.28175
16	1	17	0.00700

Figure 18 : Les fichiers genome-scores.csv

- **Le fichier rating.csv**

Toutes les évaluations sont contenues dans le fichier ratings.csv. Chaque ligne de ce fichier après la ligne d'en-tête représente une évaluation d'un film par un utilisateur, le tableau a le format suivant : Id utilisateur, Id film, rating et timestamp. Les lignes de ce fichier sont d'abord triées par userId, puis, dans user, par movieId. Les évaluations sont faites selon une échelle de 5 étoiles, avec des incréments d'une demi-étoile (0,5 étoiles - 5,0 étoiles). Les timestamps représentent les secondes depuis minuit en temps universel coordonné (UTC).

```
In [5]: # ratings
        ratings = pd.read_csv('C:/Users/info/Python/Code_Source_de_Memoire/dataset/ratings.csv')
        ratings
```

Out[5]:

	userId	movieId	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931
...	...	...	...	...
100831	610	166534	4.0	1493848402
100832	610	168248	5.0	1493850091
100833	610	168250	5.0	1494273047
100834	610	168252	5.0	1493846352
100835	610	170875	3.0	1493846415

Figure 19 : ratings.csv

➢ **Le fichier movies.csv**

Ce fichier représente les films avec leur ids, titres et genres.

```
In [6]: # movie titles
movies = pd.read_csv('C:/Users/info/Python/Code_Source_de_Memoire/dataset/movies.csv')
movies
```

Out[6]:

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy
...	...	...	...
9737	193581	Black Butler: Book of the Atlantic (2017)	Action|Animation|Comedy|Fantasy
9738	193583	No Game No Life: Zero (2017)	Animation|Comedy|Fantasy
9739	193585	Flint (2017)	Drama
9740	193587	Bungo Stray Dogs: Dead Apple (2018)	Action|Animation
9741	193609	Andrew Dice Clay: Dice Rules (1991)	Comedy

Figure 20 : Le fichier movies.csv

➢ **Le fichier genome-tags.csv**

Tous les tags sont contenus dans le fichier tags.csv. Chaque ligne de ce fichier après la ligne d'en-tête représente un tag appliqué à un film par un utilisateur et a le format suivant :

userId, movieId, tag et timestamp. Les lignes de ce fichier sont d'abord triées par userId, puis, dans user, par movieId. Les tags sont des métadonnées générées par l'utilisateur sur les films. Chaque tag est généralement un seul mot ou une courte phrase. La signification, la valeur et le but d'un tag particulier sont déterminés par chaque utilisateur. Les timestamps représentent les secondes depuis minuit en temps universel coordonné (UTC).

```
In [12]: # movie tag descriptions
genome_tags = pd.read_csv('C:/Users/info/Python/Code_Source_de_Memoire//ml-20m/genome-tags.csv')
genome_tags.head(20)
```

Out[12]:

	tagId	tag
0	1	007
1	2	007 (series)
2	3	18th century
3	4	1920s
4	5	1930s
5	6	1950s
6	7	1960s
7	8	1970s
8	9	1980s
9	10	19th century
10	11	3d
11	12	70mm
12	13	80s
13	14	9/11
14	15	aardman

Figure 21 : le fichier genome-tags.csv

3.4.2 sélectionner les films similaires par les tags

Apre l'exploration des données et après le choix d'un id utilisateur pour la recommandation comme le montre la figure suivante :

```
[6]  # id de l'utilisateur pour recommender
     id_user = 129
```

Figure 22 : id l'utilisateur

On suit les étapes suivantes :

- **Etape 1 : création d'un tableau tags pertinents**

Il y a environ 131262 films uniques et 1128 tags uniques dans l'ensemble de données du génome des tags MovieLens.

Chaque film a un score de pertinence pour chaque tag, ce qui représente plus de 10 millions de paires film-tag. Le score de pertinence va de 0 à 1.Tous les tags ne sont pas pertinents pour un film, nous devons donc conserver uniquement les tags les plus pertinents. Tout d'abord, nous pouvons classer les tags pour chaque film en fonction du score de pertinence.

Notez que les scores de pertinence sont bien supérieurs à 0,9, ce qui indique qu'il s'agit de tags très pertinents.

```
[7]  #merge pour trouver les relevence et title des films
     mv_tags = genome_scores.merge(genome_tags, on = 'tagId').merge(movies,on = 'movieId').drop(['genres'],axis=1)
     mv_tags.head(10)
```

	movieId	tagId	relevance	tag	title
0	1	1	0.02500	007	Toy Story (1995)
1	1	2	0.02500	007 (series)	Toy Story (1995)
2	1	3	0.05775	18th century	Toy Story (1995)
3	1	4	0.09675	1920s	Toy Story (1995)
4	1	5	0.14675	1930s	Toy Story (1995)
5	1	6	0.21700	1950s	Toy Story (1995)
6	1	7	0.06700	1960s	Toy Story (1995)
7	1	8	0.26275	1970s	Toy Story (1995)
8	1	9	0.26200	1980s	Toy Story (1995)
9	1	10	0.03200	19th century	Toy Story (1995)

Figure 23 : tableau tag

- **Etape 2 : trier les tags selon la pertinence :**

Dans cette partie, on fait le tri des tags par ordre décroissant selon la pertinence.

```
#methode ranking pour relevance
mv_tags['rel_rank'] = mv_tags.groupby("movieId")["relevance"].rank(method = "first", ascending = False)
mv_tags
```

	movieId	tagId	relevance	tag	title	rel_rank
0	1	1	0.02500	007	Toy Story (1995)	924.0
1	1	2	0.02500	007 (series)	Toy Story (1995)	925.0
2	1	3	0.05775	18th century	Toy Story (1995)	660.0
3	1	4	0.09675	1920s	Toy Story (1995)	507.0
4	1	5	0.14675	1930s	Toy Story (1995)	379.0

Figure 24 : utilisant méthode Rank

Un exemple avec film Jumaji (1995) qui représente la liste des 100 meilleurs tags.

```
# un les tags tops pour les films
mv_tags[mv_tags.title == "Jumanji (1995)"][['movieId','relevance','tag','title','rel_rank']].sort_values(by = 'relevance', ascending = False).head(10)
```

	movieId	relevance	tag	title	rel_rank
1156	2	0.98100	adventure	Jumanji (1995)	1.0
1711	2	0.96700	jungle	Jumanji (1995)	2.0
1331	2	0.96425	children	Jumanji (1995)	3.0
1715	2	0.95850	kids	Jumanji (1995)	4.0
2078	2	0.93475	special effects	Jumanji (1995)	5.0
1504	2	0.93400	fantasy	Jumanji (1995)	6.0
1189	2	0.91450	animals	Jumanji (1995)	7.0
1542	2	0.87050	fun movie	Jumanji (1995)	8.0
1330	2	0.87025	childhood	Jumanji (1995)	9.0
1501	2	0.86925	family	Jumanji (1995)	10.0

Figure 25 : list tags

- **Etape 3 : sélection des tags**

On sélectionne les 100 premiers tags (meilleurs) pour chaque film en les mettant dans une liste.

```
#100 utilise les tags plus relevance pour chaque film
mv_tags_list = mv_tags[mv_tags.rel_rank <= 100].groupby(['movieId','title'])['tag'].apply(lambda x: ','.join(x)).reset_index()
mv_tags_list['tag_list'] = mv_tags_list.tag.map(lambda x: x.split(','))

#list des tags
pd.set_option('display.max_colwidth', -1)

mv_tags_list.loc[mv_tags_list.title == 'Jumanji (1995)', ['movieId','title','tag_list']]
```

	movieId	title	tag_list
1	2	Jumanji (1995)	[action, action packed, adaptation, adapted from:book, adventure, alter ego, animal movie, animals, bad cgi, based on a book, based on a video game, based on book, big budget, bullying, catastrophe, cgi, chase, childhood, children, clever, comedy, computer animation, computer game, cool, creativity, cutel, death, destiny, dialogue, dinosaurs, dynamic cgi action, effects, entertaining, excellent, exciting, family, fantasy, fantasy world, fast paced, feel-good, fight scenes, first contact, fun, fun movie, good, good action, goofy, great, great ending, great movie, happy ending, heartwarming, high fantasy, hunting, intense, island, jungle, kids, kids and family, life philosophy, light, lions, magic, mentor, monkey, natural disaster, nostalgic, original, original plot, oscar (best effects - visual effects), oscar (best supporting actress), oscar winner, pg, pg-13, predictable, redemption, runaway, saturn award (best special effects), scary, scifi, sentimental, silly, silly fun, special effects, spiders, story, supernatural, suspense, technology, time, time travel, toys, transformation, underdog, video game, video games, videogame, visual, visually stunning, witches]

Figure 26 : liste des 100 meilleurs tags

Ensuite, nous confirmons dans le tableau ci-dessous que les tags les mieux classés pour un film ont tendance à avoir des scores de pertinence médians plus élevés. Les tags au 1er rang pour un film ont un score de pertinence médian de près de 1. Nous pouvons voir que le score de pertinence médian diminue progressivement au fur et à mesure que nous descendons au 50ème rang.

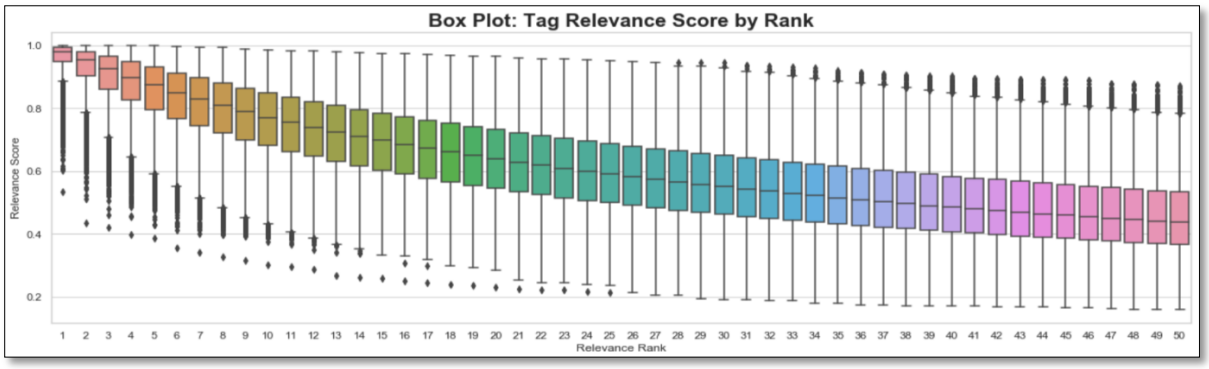

Figure 27 : graph tags les mieux classés

Le graphique ci-dessous montre la variation en pourcentage du score de pertinence médian au fur et à mesure que nous passons des tags du 1er au 100e rang. Nous voyons un point d'inflexion autour du 50e rang lorsque le score de pertinence commence à devenir plus stable. Ainsi, nous pouvons choisir N = 50 comme nombre raisonnable de tags à conserver pour chaque film.

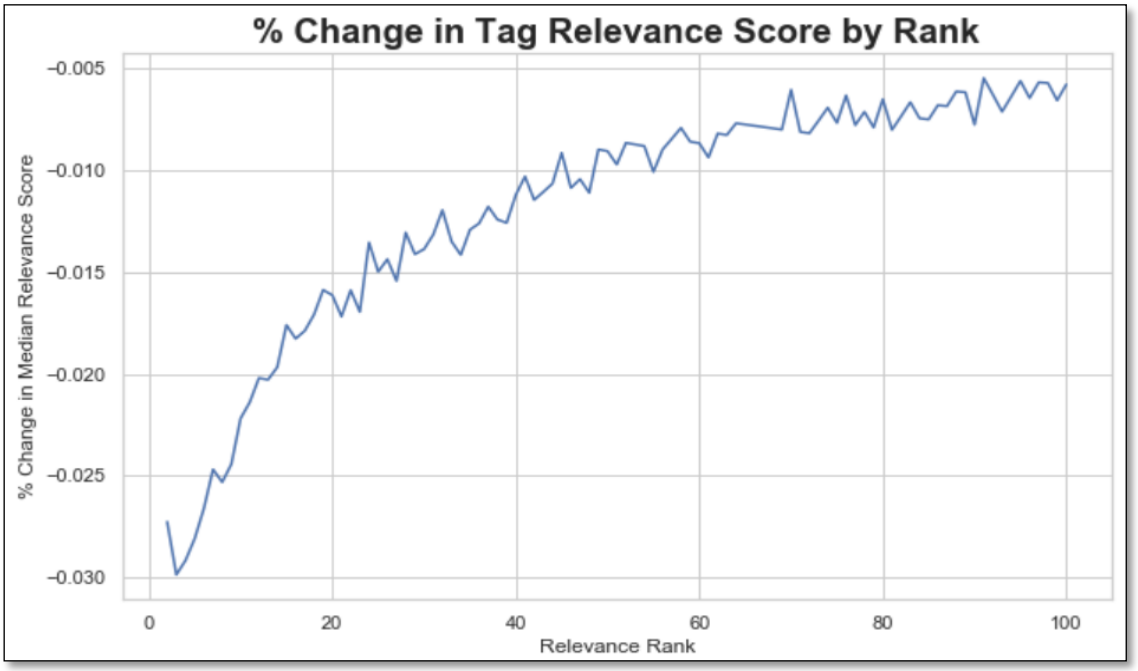

Figure 28 : graph score de pertinence médian

Pour chaque film utilise les tops tag pour recommandation qui trouver les films similaire.

```
In [27]: pd.reset_option('display.max_colwidth')

         target_movie = 'Toy Story (1995)'

         target_tag_list = mv_tags_list[mv_tags_list.title == target_movie].tag_list.values[0]
         mv_tags_list_sim = mv_tags_list[['movieId','title','tag_list','tag']]
         mv_tags_list_sim.head(10)
```

Out [27]:

	movieId	title	tag_list	tag
0	1.0	Toy Story (1995)	[3d, action, adventure, affectionate, animal m...	3d,action,adventure,affectionate,animal movie,...
1	2.0	Jumanji (1995)	[action, action packed, adaptation, adapted fr...	action,action packed,adaptation,adapted from b...
2	3.0	Grumpier Old Men (1995)	[adaptation, adventure, bad ending, beautiful ...	adaptation,adventure,bad ending,beautiful scen...
3	4.0	Waiting to Exhale (1995)	[adaptation, adapted from book, adultery, bad ...	adaptation,adapted from book,adultery,bad acti...
4	5.0	Father of the Bride Part II (1995)	[adaptation, bad plot, bad sequel, beautiful s...	adaptation,bad plot,bad sequel,beautiful scene...
5	6.0	Heat (1995)	[action, action packed, amazing cinematography...	action,action packed,amazing cinematography,at...
6	7.0	Sabrina (1995)	[adaptation, based on a play, beautiful, beaut...	adaptation,based on a play,beautiful,beautiful...
7	8.0	Tom and Huck (1995)	[action, action packed, adaptation, adapted fr...	action,action packed,adaptation,adapted from b...
8	9.0	Sudden Death (1995)	[action, action packed, adaptation, adapted fr...	action,action packed,adaptation,adapted from g...
9	10.0	GoldenEye (1995)	[007, 007 (series), action, action packed, adv...	007,007 (series),action,action packed,adventur...

Figure 29 : les films avec top tag

- **Etape 4 : nettoyage des tags**

Dans cette étape, on fait un nettoyage sur les tags sélectionnés en supprimant les ponctuations et les chiffre indésirables.

```
[ ]  # corpus pour les films tags
     mv_tags_corpus = mv_tags_list.tag.values

                                                    + Code    + Text

[ ]  import nltk
     nltk.download('stopwords')

     [nltk_data] Downloading package stopwords to /root/nltk_data...
     [nltk_data]   Unzipping corpora/stopwords.zip.
     True

[ ]  # netoiyer les tags
     stop_words = stopwords.words('english')

     # tokenize document and clean
     def word_tokenize_clean(doc):

       # split into lower case word tokens
         tokens = word_tokenize(doc.lower())

       # remove tokens that are not alphabetic (including punctuation) and not a stop word
         tokens = [word for word in tokens if word.isalpha() and not word in stop_words]

         return tokens
```

Figure 30 : nétoyer les tags

```
[ ] import nltk
    nltk.download('punkt')

    [nltk_data] Downloading package punkt to /root/nltk_data...
    [nltk_data]   Unzipping tokenizers/punkt.zip.
    True

[ ] # preprocess corpus of movie tags before feeding it into Doc2Vec model
    mv_tags_doc = [TaggedDocument(words=word_tokenize_clean(D),tags=[str(i)]) for i, D in enumerate(mv_tags_corpus)]
```

Figure 31 : liste des tags

➢ Etape 5 : transformation des tags en vecteur

On utilise Doc2Vec pour Transformer les caractères comme en vecteurs, en créant un modèle qui se base sur max_epochs =30 :

```
[ ] # instantiate Doc2Vec model
    max_epochs = 30
    vec_size = 20
    alpha = 0.025
    # Create doc2vec model.
    model = Doc2Vec(size=vec_size,alpha=alpha, min_alpha=0.00025, min_count=1,dm=0)

    # paragraph vector distributed bag-of-words (PV-DBOW)
    model.build_vocab(mv_tags_doc)

    # train Doc2Vec model
    # stochastic (random initialization), so each run will be different unless you specify seed
    print('Epoch', end = ': ')
    for epoch in range(max_epochs):

        print(epoch, end = ' ')

        model.train(mv_tags_doc, total_examples=model.corpus_count, epochs=model.epochs)

    # decrease the learning rate
    model.alpha -= 0.0002

    # fix the learning rate, no decay
    model.min_alpha = model.alpha

    Epoch: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29

    mv_tags_vectors = model.docvecs.vectors_docs
    mv_tags_vectors.shape
```

Figure 32 : Doc2Vec

- **Etape 6 : sélection les films taguées par l'utilisateur :**

On sélectionne les films tagués par un utilisateur donnée :

```
[ ]  user= [id_user]
     df = pd.DataFrame(user, columns= ['userId'])
     df1 = pd.merge(tags,df)
     userId = pd.merge(df1,movies).drop(['movieId','genres'],axis=1)
     userId.head(10)
```

	userId	tag	title
0	129	Adam Sandler	Happy Gilmore (1996)
1	129	golf	Happy Gilmore (1996)
2	129	sports	Happy Gilmore (1996)
3	129	Arnold Schwarzenegger	Jingle All the Way (1996)
4	129	christmas movie	Jingle All the Way (1996)
5	129	Xmas theme(?)	Jingle All the Way (1996)
6	129	Adam Sandler	Waterboy, The (1998)
7	129	American football	Waterboy, The (1998)
8	129	football	Waterboy, The (1998)
9	129	sport:American football	Waterboy, The (1998)

Figure 33 : films qui tags

- **Etape 7 : trouver les films similaires à base de tags**

Utilise la méthode cosinus pour trouver la similarité entre les films selon le code suivant :

```
[ ]  #movies recommender
     mv = "Carlito's Way (1993)"

     mv_index = mv_tags_list[mv_tags_list["title"] == mv].index.values[0]

     sims = model.docvecs.most_similar(positive = [mv_index], topn = 10)

     # les films similaire
     for i, j in sims:
         print(mv_tags_list.loc[int(i), "title"].strip())
```

```
Training Day (2001)
American Gangster (2007)
Deep Cover (1992)
Thief (1981)
Heat (1995)
Angels with Dirty Faces (1938)
Rounders (1998)
Goodfellas (1990)
Across 110th Street (1972)
White Heat (1949)
```

Figure 34 : les films similaires

En tenant compte des films tagués par l'utilisateur, on peut sélectionner les films similaires à ces films selon un seuil donné.

```
# les movies tags pour user
user_movies = user_tag

user_movie_vector = np.zeros(shape = mv_tags_vectors.shape[1])
#les vecteur
for mv in user_movies:

    mv_index = mv_tags_list[mv_tags_list["title"] == mv].index.values[0]

    user_movie_vector += mv_tags_vectors[mv_index]

    user_movie_vector /= len(user_movies)


print('Movie Recommendations :')
```

Figure 35 : les films recommandent

3.4.3 sélectionner les films similaires par l'évaluation

- **Etape 1 : La création d'un tableau (films-évaluation)**

Dans cette partie, on fusionne les fichiers movies.csv et ratings.csv dans un seul fichier nommé m_ratings.

```
[ ]  #trouver les titres pour movies
     m_ratings = pd.merge(movies,ratings).drop(['genres'],axis=1)
     m_ratings
```

	movieId	title	userId	rating
0	1	Toy Story (1995)	1	4.0
1	1	Toy Story (1995)	5	4.0
2	1	Toy Story (1995)	7	4.5
3	1	Toy Story (1995)	15	2.5
4	1	Toy Story (1995)	17	4.5
...	...	...	...	...
77992	131023	That Sugar Film (2014)	119	4.5
77993	131098	Saving Santa (2013)	89	5.0
77994	131104	The Brain (1969)	89	4.5

Figure 36 : un tableau ratings

➢ **Etape 2 : matrice utilisateur films (rating)**

Dans cette partie, on va créer une matrice (id user - films titles).

```
#matrice
userRatings = m_ratings.pivot_table(index=['userId'],columns=['title'],values='rating')
userRatings.head(10)
```

title	'Hellboy': The Seeds of Creation (2004)	'Round Midnight (1986)	'Salem's Lot (2004)	'Til There Was You (1997)	'burbs, The (1989)	'night Mother (1986)	(500) Days of Summer (2009)	*batteries not included (1987)	...All the Marbles (California Dolls, The) (1981)
userId									
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Figure 37 : matrice de rating

On fait un petit prétraitement en remplaçant les valeurs NAN par des 0.

```
# remlplacer les NAN avec 0
userRatings.fillna(0, inplace=True)
print("After: ",userRatings.shape)
```

```
After:  (500, 7781)
```

Figure 38 : remplace les NAN avec des 0

➢ **Etape 3 : matrice de corrélation de Pearson**

On utilise la méthode de corrélation de Pearson pour trouver la similarité entre les films selon les évaluations.

```
[ ]  #matrix de correlation avec method pearson trouver les films similaire
     corrMatrix = userRatings.corr(method='pearson')
     corrMatrix.head(5)
```

title	'Hellboy': The Seeds of Creation (2004)	'Round Midnight (1986)	'Salem's Lot (2004)	'Til There Was You (1997)	'burbs, The (1989)	'night Mother (1986)
title						
'Hellboy': The Seeds of Creation (2004)	1.000000	0.706398	-0.002004	-0.002752	-0.006818	-0.002004
'Round Midnight (1986)	0.706398	1.000000	-0.002837	-0.003896	0.209720	-0.002837

Figure 39 : méthode similaire

Ce code représente la fonction qui trouver la matrice de similaire entre les films qui évalue avec utilisateur et tous les films.

```
def get_similar(movie_name):
        similar_ratings = corrMatrix[movie_name]

        similar_ratings = similar_ratings.sort_values(ascending=False)

        return similar_ratings
```

Figure 40 : matrices des similar

- **Etape 4 : sélectionner les films évalués par l'utilisateur**

  On sélectionne les films qui ont été évalués par l'utilisateur avec une évaluation supérieur ou égale à 4.

```
# trouver le film qui ratings avec utilisateur
user= [id_user]
df = pd.DataFrame(user, columns= ['userId'])
df1 = pd.merge(ratings,df)
userId = pd.merge(df1,movies).drop(['movieId','genres'],axis=1)
userId.head(10)
```

	userId	rating	title
0	129	3.5	Seven (a.k.a. Se7en) (1995)
1	129	4.0	Braveheart (1995)
2	129	3.5	Congo (1995)
3	129	3.5	Net, The (1995)
4	129	4.0	Waterworld (1995)
5	129	4.0	Goofy Movie, A (1995)
6	129	3.5	Interview with the Vampire: The Vampire Chronicles (1994)
7	129	4.0	Star Wars: Episode IV - A New Hope (1977)
8	129	3.5	Outbreak (1995)
9	129	4.5	Stargate (1994)

Figure 41 : ratings pour user

- **Etape 5 : les films similaires**

Sélectionne les films similaires base sur l'évaluation des films par l'utilisateur.

```
[ ]  #trouver les films similaire
     # list les films rate avec user
     movie = lis_mv_us
     similar_movies = pd.DataFrame()
     #remplir les films par order
     for movie in movie:
          similar_movies = similar_movies.append(get_similar(movie))

     similar_movies.sum().sort_values(ascending=False).head(10)
```

```
X2: X-Men United (2003)            31.520400
Matrix Reloaded, The (2003)        30.540014
X-Men: The Last Stand (2006)       30.491686
Batman Begins (2005)               30.482998
Spider-Man (2002)                  30.441184
Minority Report (2002)             29.862105
Signs (2002)                       29.819347
Fifth Element, The (1997)          29.791197
Shaun of the Dead (2004)           29.748554
War of the Worlds (2005)           29.490457
dtype: float64
```

Figure 42 : les films similaires

- **Etape 6 : sélectionner les films évalués par le seuil**

On choisit les films similaires par rate avec seuil (somme de similaire).

```
[ ]  # semilaire pour les ratings
     set2 = similar_movies.sum().sort_values(ascending=False)
     # trasfer pour dataframe
     set3 = pd.DataFrame(set2, columns= ['rating'])
     #creer un seuil
     rating_seuil = set3[set3['rating']>=2].index
     # tromsfer les films en list
     mv_rate = rating_seuil.tolist()
     print (mv_rate[:10])
```

```
['X2: X-Men United (2003)', 'Matrix Reloaded, The (2003)',
```

Figure 43 : seuil par ratings

3.4.5 La recommandation par l'intersection des deux sous-ensembles

Le cœur de notre approche est basé sur l'intersection des films similaires à base de tags et à base d'évaluation, elle est comme suit :

On fixe le seuil de la recommandation puis on va recommander 10 films les 5 premiers se trouvent dans l'intersection des deux ensembles, et les 5 autres sont sélectionner aléatoirement d'après les films recommander avec les tags. Si les nombre des films recommandés à l'intérieur de l'intersection est inférieur à 5, on rajoute le reste d'après les tags. Si les utilisateurs ont évalué les films mais n'ont pas tagué, on recommande les films selon l'évaluation. Si l'utilisateur est nouveau, on le recommande des films récents.

```
[ ]             # les movies pour les tags
                if movie_sim not in mv_rate:

                    b.append(movie_sim)

    import numpy
    def selectRandom(b):

        return numpy.random.choice(b, 5)

    aleatoire_list_tag = list(selectRandom(b))
    # 10 les films recomender
    r = len(int_mv)
    if  r >= 5 :
      mv_rec = int_mv[:5]+aleatoire_list_tag

    elif r < 5 :
        j = 5-r
        nb = j+5
        mv_rec = int_mv[:5]+b[:nb]
    l = len(mv_rec)
    if l == 0 :
      print (dff)
    else :
      print (mv_rec)

    Movie Recommendations :
    ['Anchorman: The Legend of Ron Burgundy (2004)', 'Talladega Nights: The Bal
```

Figure 44: List films recommandation

3.4.6 Explication de la recommandation

Pour explique notre recommandation, on affiche à l'utilisateur un graphe 2D sur lequel on montre la position des films recommandés et les films sur lesquels on s'est basé.

Les films en noir se sont des films sur lesquels on s'est basé (films tagués par l'utilisateur avec une grande pertinence), les films en vert sont les films évalués par l'utilisateur avec une bonne note et les films en rouge se sont les films recommandés à l'utilisateur.

Sur ce graphe, on peut voir que les films recommandés sont proches les uns des autres. Il y a une légère divergence qui est due à la diversité.

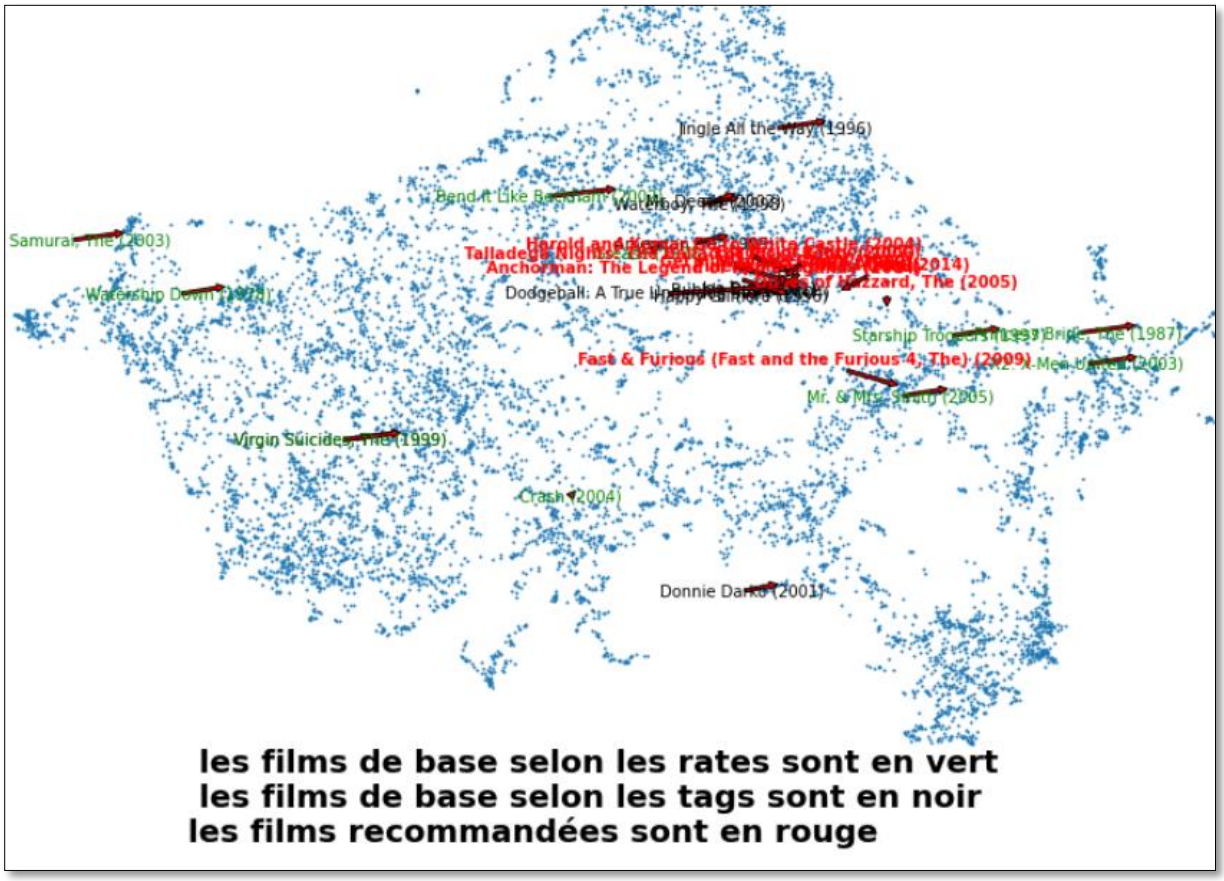

Figure 45 : graph 2D représente les films qui recommander

Afin de donner une bonne vue à l'utilisateur, on lui affiche aussi un schéma représenté par trois cercles, le premier cercle en bleu représente les films similaires par les tags, le deuxième cercle qui est en rouge représente les films similaires par les rates et le troisième cercle qui est l'intersection représente les films similaires par les tags et les évaluations.

Figure 46 : cercle pour explique

Dans figure 47 en afficher les 5 premier de films qui recommandé avec tags, rating et intersection les plus tops.

3.6 Interprétation des résultats

Dans cette partie, nous représentons l'évaluation expérimentale de notre application, pour illustrer l'efficacité de notre algorithme de filtrage base sur contenu.

Pour évaluer notre système de recommandation on calcule :

La diversité : calculée par la formule (1- valeur similaire).

La pertinence : représente la valeur des films qui sont recommandé avec la similarité supérieur à 0.5 ,0.2, 0.8 voilà un exemple qui calcule la diversité et la pertinence :

```
[154] # calcul les devirsite
      df4 = df2['valeur_similaire'].tolist()
      som = 0
      p = 0
      for i in df4 :
          som +=i
          if i >= 0.5:
              p += 1
      som = som/10
      print ('devirsite',1-som)
      #precision
      p= p/10
      print ('pertinence',p)

      devirsite 0.2013563990592957
      pertinence 1.0
```

Figure 47 : calcul diversité et précision

L’exemple ci-dessous illustre une recommandation à l’utilisateur avec id = 121.

- Première colonne : représente les films tagués par l’utilisateur.
- Deuxième colonne : représente les films évalués par l’utilisateur.
- Troisième colonne : représente les films similaires à base de tags.
- Quatrième colonne : représente les films recommandés.

```
#les films qui recommender final
d1 = pd.DataFrame(mv_rec, columns= ['movies_recommender'])
#les films qui recommender avec  tags
d3 = b[:10]
d2 = pd.DataFrame(d3, columns= ['movies_recommender_tag'])
# les films qui tag pour utilisateur
d4 = pd.DataFrame(user_movies, columns= ['mv_tag_user'])
# les films qui rating pour utilisateur
d5 = pd.DataFrame(lis_mv_us , columns= ['mv_rating_user'])
#tableu de affichage
tab_user = pd.concat([d4,d5,d2,d1] , axis=1, sort=False)
tab_user
```

	mv_tag_user	mv_rating_user	movies_recommender_tag	movies_recommender
0	Happy Gilmore (1996)	Starship Troopers (1997)	Blades of Glory (2007)	Anchorman: The Legend of Ron Burgundy (2004)
1	Jingle All the Way (1996)	Crash (2004)	22 Jump Street (2014)	Talladega Nights: The Ballad of Ricky Bobby (2006)
2	Waterboy, The (1998)	Last Samurai, The (2003)	Get Hard (2015)	Wayne's World (1992)
3	American Pie (1999)	Virgin Suicides, The (1999)	Waiting... (2005)	Harold and Kumar Go to White Castle (2004)
4	Virgin Suicides, The (1999)	Princess Bride, The (1987)	Accepted (2006)	Pineapple Express (2008)
5	Joe Dirt (2001)	Mr. & Mrs. Smith (2005)	BASEketball (1998)	Fast & Furious (Fast and the Furious 4, The) (2009)
6	Bubble Boy (2001)	Watership Down (1978)	Other Guys, The (2010)	Dukes of Hazzard, The (2005)

Figure 48 : tableaux des films

Le tableau suivant représente l’évaluation de notre système avec différents utilisateurs.

Utilisateur	Seuil de similarité	Diversité	Pertinence	
			P	Valeur
User 121	0.8	0.181	0.8	1.0
			0.5	1.0
	0.5	0.3359	0.8	0.4
			0.5	1.0
	0.2	0.349	0.8	0.1
			0.5	0.9
User 96	0.8	0.1308	0.8	1.0
			0.5	1.0
	0.5	0.26129	0.8	0.3

			0.5	1.0
	0.2	0.2575	0.8	0.1
			0.5	1.0

Figure 49 : tableaux évolution

On observe que :

1. Quand le seuil est grand (0.8) et p = 0.8, la diversité est trop petite et la pertinence est grande,
2. Quand le seuil est grand (0.8) et la valeur de similarité est 0.5 (p) la diversité est trop petite et la pertinence diminue,
3. Quand le seuil est égale à 0.5 et la valeur de similarité est 0.8 (p) la diversité augmente et la pertinence diminue,
4. Quand le seuil est à 0.5 et la valeur de similarité est 0.5 (p) la diversité se stabilise et la pertinence augmente,
5. Quand le seuil est petit (0.2) et la valeur de similarité est 0.8 (p) la diversité augmente et la pertinence diminue,
6. Quand le seuil est petit (0.2) et la valeur de similarité est 0.5 (p) la diversité et la précision augmente,

Remarque : La valeur de pertinence 0.9 avec la diversité 0.4 est considérer comme le résultat préférer. On a trouvé ce résultat dans le 6eme cas. Quand le seuil diminue la diversité augmente.

Résultat : Nous trouvons que la meilleure étude est le seuil 0.5 et la valeur de similarité 0.5

Résultat final : La meilleure mesure de l'efficacité d'un algorithme de recommandation et de la pertinence des suggestions c'est finalement la satisfaction de l'utilisateur, qui n'est pas toujours facile à bien identifier.

3.7 Conclusion

Dans ce chapitre, nous avons créé un système de recommandation basé sur les tags. en calculant la similarité sur les tags et les rates,

Afin d'évaluer la performance des recommandations des films avec l'utilisateur nous avons calculé la diversité et la pertinence.

CONCLUSION ET PERSPECTIVES

Les systèmes de recommandation automatique sont devenus, à l'instar des moteurs de recherche, un outil incontournable pour tout site Web focalisé sur un certain type d'articles disponibles dans un catalogue riche, que ces articles soient des objets, des produits culturels (livres, films, morceaux de musique, etc.), des éléments d'information (news) L'objectif de ces systèmes est de sélectionner, dans leur catalogue, les items les plus susceptibles d'intéresser un utilisateur particulier ont répertorié un vaste ensemble de systèmes de recommandation pour différents domaines applicatifs, dans des contextes académiques et industriels.

Le travail présenté dans ce mémoire rentre dans le cadre du contexte de la recommandation system avec explication. Nous avons donné une vue générale sur ce domaine en introduisant la notion de recommandation avec explication pour recommandation en Calculant des similarités.

Dans une perspective d'avenir, on a résumé plusieurs nouvelles perspectives possibles sur la recherche de recommandations explicables :

- On souhaite développer notre système tel qu'il doit créer la table d'évaluation automatiquement.
- Dans le futur, on va essayer d'utiliser la recommandation hybride dans notre système.
- On va essayer de recommander aux nouveaux utilisateurs des films basés sur les informations de leur profiles.
- Fusionner les techniques de recommandation avec d'autres champs de recherche.

REFERENCES

[01] *Système de recommandation*. (s.d.). Récupéré sur wikipedia: https://fr.wikipedia.org/wiki/Syst%C3%A8me_de_recommandation

[02] Fatiha, D. d. (Proposition d'un système de recommandation). *Proposition d'un système de recommandation.* Université Saad DAHLAB - Blida .

[05] *ontent based recommende*. (s.d.). Récupéré sur kdnuggets: https://www.kdnuggets.com/2019/11/content-based-recommender-using-natural-language-processing-nlp.html

[06] Rich, E. 1979. User Modeling via Stereotypes. Cognitive Science 3(4):329–354.

[07] Rocchio, J. J. 1971. Relevance Feedback in Information Retrieval. In The SMARTRetrievalSystem:ExperimentsinAutomaticDocumentProcessing,ed.G.Salto n,313–323. Englewood Cliffs, NJ: Prentice-Hall.

[08] Meryem, M. K. (14 Juin 2015). *Système de recommandation desservices web sémantiques.*Université Abou Bakr Belkaid– Tlemcen.

[09] *Les systèmes de recommandation : une catégorisation*. (s.d.). Récupéré sur interstices.info:https://interstices.info/les-systemes-de-recommandation-categorisation/

[10] *A la découverte des systèmes de recommandation*. (s.d.). Récupéré sur invivoo: https://www.invivoo.com/systeme-de-recommandation/

[11] Elsa.Negre. (25/06/2012). *système de recommandation.* Université Paris-Dauphine.

[12] *Filtrage_collaboratif*. (s.d.). Récupéré sur wikipedia: https://fr.wikipedia.org/wiki/Filtrage_collaboratif

[13] Fayrouz Soualah-Alila. CAMLearn: Une Architecture de Système de Recommandation Sémantique Sensible au Contexte. Application au Domaine du M-Learning. Informatique [cs]. Université de Bourgogne, 2015. Français.Fftel 01238233v2f

[14] K. Goldberg, T. Roeder, D. Gupta, C. Perkins Eigentaste: a constant time collaborative filtering algorithm Inform Retrieval J, 4 (2) (2001)

[15] Claypool M, Gokhale A, Miranda T, Murnikov P, Netes D, Sartin M. Combining content- based and collaborative filters in an online newspaper. In: Proceedings of ACM

SIGIR workshop on recommender systems: algorithms and evaluation, Berkeley, California; 1999.

[16] B.M. Sarwar, J.A. Konstan, J.L. Herlocker, B. Miller, J.T. Riedl Using filtering agents to improve prediction quality in the grouplens research, collaborative filtering system Proceedings of the ACM conference on computer supported cooperative work, ACM, New York (NY, USA) (1998), pp. 345-354

[17] M. Drosou, E. Pitoura Search result diversification SIGMOD Rec, 39 (1) (2010), pp.41-47

[18] M. Papagelis, D. Plexousakis Qualitative analysis of user-based and item-based prediction algorithms for recommendation agents Int J Eng Appl Artif Intell, 18 (4) (2005), pp.781-789

[19] Nguyen, A.-T. (n 17 Jan 2009). *Un nouveau système de filtrage collaboratif*. https://tel.archives-ouvertes.fr/tel-00353945/document: /tel.archives-ouvertes.fr

[20] *Créer un système de recommandation dans TensorFlow : présentation*. (s.d.). Récupéré sur tensorflow: https://cloud.google.com/architecture/recommendation-system-tensorflow-overview?hl=fr

[23] *recommender systems*. (s.d.). Récupéré sur kaggle: https://www.kaggle.com/prashant111/recommender-systems-in-python

[26] *doc2vec*. (s.d.). Récupéré sur heartbeat: https://heartbeat.fritz.ai/getting-started-with-doc2vec-2645e3e9f137

[27] multi-class-text-classification-with-doc2vec-logistic-regression. (s.d.). Récupéré sur datascience: https://towardsdatascience.com/multi-class-text-classification-with-doc2vec-logistic-regression-9da9947b43f4

[28] *similarity-in-python-using-doc2vec*. (s.d.). Récupéré sur kanok: https://kanoki.org/2019/03/07/sentence-similarity-in-python-using-doc2vec/

[29] Tintarev, N. (January 2007). *Explanations of recommendations.*Delft University of Technology.

[30] Tintarev, N. (January 2007). *Explanations of recommendations.*Delft University of Technology

[31] *Tags*. (s.d.). Récupéré sur researchgate: https://www.researchgate.net/figure/Tags-as they-appear-on-the-MovieLens-movie-details-screen_fig3_22102301

[32] (2020-2021). *Etude des systèmes de recommandations et mise en pratique des algorithmes.*HEC-Ecole de gestion de l'Université de Liège.

[33] *Python_(langage)*. (s.d.). Récupéré sur wikipedia: https://fr.wikipedia.org/wiki/Python_(langage)

[34] *Jupyter*. (s.d.). Récupéré sur wikipedia: https://fr.wikipedia.org/wiki/Jupyter

[35] *jupyter*. (s.d.). Récupéré sur jupyte: https://jupyter.org/?fbclid=IwAR0xmZkyAdPX2R-ZK84Gv6ZqSzVj3my8GyhNpTooZhl7G-MvJmCRh1Zvgoo

[36] *Anaconda_(Python_distribution)*. (s.d.). Récupéré sur wikipedia: https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)

[37] (s.d.). Récupéré sur anaconda: https://www.anaconda.com/products/individual-b?fbclid=IwAR2sBCawO_62pO6Nf7AKiPopyJc0IA4_T6MDbbr7DAJVZnWYoTKsY0q6N40