

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique  
Université des Sciences et de la Technologie Houari Boumediene  
Faculté d'électronique et d'informatique  
Département d'informatique



# Mémoire de Master

Domaine Informatique

Option : Systèmes informatiques intelligents (SII)

Thème

Recommandation par filtrage collaboratif basé sur  
un clustering optimisé

**Présenté par :**  
- BELLAHMER Meziane

**Sujet proposé par :**  
- Dr. Lamia BERKANI

**Devant le jury composé de :**

- Prof. A.ABDELLI  
- Dr. Z.BENBAZIZ

Président  
Membre

Projet N : 063/2018

# Remerciements

Je tiens à exprimer toute ma gratitude, qui va en premier lieux à mes parents, eux qui m'ont tant soutenu dans mes études et qui ont tant donné pour me voir réussir un jour.

Je remercie Dieux, pour m'avoir donné la santé, la force et la motivation nécessaires pour mener à bien ce travail. Un remerciement particulier pour mon encadreur, Dr L. Berkani, pour son écoute, sa confiance et ses précieux conseils qui m'ont permis de mener à bien mon projet.

Mes remerciements s'adressent également aux membres du jury qui m'ont fait l'honneur d'accepter d'évaluer ce travail. Je remercie le Prof. A. Abdelli d'avoir bien voulu présider ce jury. Je remercie également Mme Z. Benbaziz d'avoir accepté d'examiner ce travail.

# Dédicaces

Je dédie ce modeste travail à mon père, ma mère, ma famille, ainsi que mes amis, qui furent présents à mes coté lors de nombreuses épreuves et m'ont toujours apporté un soutien incontesté. Je souhaiterai également transmettre un grand remerciement à tous ces enseignants qui ont intervenu dans mon parcours d'élève ou d'étudiant, et auprès desquelles j'aurai tant appris et tant évolué, et qui ont fais que j'acquiers un bagage assez important afin de parvenir un jour à effectuer un travail comme celui-ci.

J'aimerai en outre transmettre une pensée toute particulière, à nos frères palestiniens, qui vivent des heures sombres et qui connaissent depuis des décennies, le chaos et la tourmente. Un soutient également, à tout ces peuples opprimés, victimes des caprices d'une poignée d'hommes et que le monde refuse de dénoncer ; à ces chercheurs et à ces scientifiques, pour leur travail sans relâche dans le but d'améliorer la condition humaine et non pour des ambitions destructrices.

*"Beaucoup de science découvre à l'homme sa vaste ignorance."*

Edward Young

# Sommaire

<b>I</b>	<b>Etat de l'art</b>	<b>3</b>
<b>1</b>	<b>Recommandation personnalisée</b>	<b>4</b>
1.1	Introduction . . . . .	4
1.2	Présentation des systèmes de recommandation . . . . .	5
1.3	Différentes approches pour la recommandation . . . . .	5
1.3.1	Filtrage cognitif (ou filtrage basé sur le contenu) . . . . .	6
	Avantages et inconvénients des systèmes basés sur le contenu	8
1.3.2	Filtrage collaboratif . . . . .	9
	Filtrage collaboratif basé mémoire . . . . .	10
	Filtrage collaboratif basé modèle . . . . .	13
	Filtrage hybride . . . . .	15
1.4	Profil utilisateur . . . . .	16
1.5	Conclusion . . . . .	17
<b>2</b>	<b>Classification et optimisation</b>	<b>19</b>
2.1	Qu'est ce que la classification ? . . . . .	19

2.2	Classification supervisée . . . . .	20
2.2.1	Les arbres de décisions . . . . .	21
2.2.2	Les classifieurs bayésiens . . . . .	22
2.2.3	Entre autres . . . . .	25
	SVM . . . . .	25
	Classifieur KNN . . . . .	25
2.3	Clustering . . . . .	26
2.3.1	Les différents types de clustering . . . . .	27
	Clustering par partitionnement(partionning-based clustering) . . . . .	27
	Clustering hiérarchique(hierarchical clustering) . . . . .	27
	Clustering basé sur la densité (density-based method) . . . . .	28
	L’approche basée sur la grille (grid-based method) . . . . .	29
2.3.2	Quelques algorithmes de clustering . . . . .	29
	K-means . . . . .	29
	K-Medoids . . . . .	30
	DBSCAN . . . . .	33
2.3.3	Le clustering dans les systèmes de recommandation . . . . .	33
2.4	Méta-heuristiques pour l’optimisation . . . . .	34
2.4.1	Les algorithmes génétiques . . . . .	34
2.4.2	Optimisation par colonie de fourmis . . . . .	35
<b>II</b>	<b>Contribution</b>	<b>40</b>
<b>3</b>	<b>Proposition d’une approche de recommandation</b>	<b>41</b>
3.1	Introduction . . . . .	41

3.2	Description de notre approche de recommandation . . . . .	41
3.2.1	Voisinage et clustering . . . . .	44
	Formalisation de l'approche de clustering . . . . .	44
	Algorithme et illustration . . . . .	44
3.2.2	Optimisation du clustering . . . . .	47
	Adaptation de la méta-heuristique . . . . .	47
	Algorithme et illustration . . . . .	52
	Un seul critère d'arrêt . . . . .	53
	Modification du concept de transition . . . . .	53
	Choix de la solution . . . . .	53
	La prédiction . . . . .	53
	La fonction de prédiction . . . . .	54
	Le fonction de corrélation . . . . .	54
3.3	Conclusion . . . . .	54
<b>4</b>	<b>Implémentation et évaluation</b>	<b>56</b>
4.1	Introduction . . . . .	56
4.2	Architecture technique du système . . . . .	57
4.3	Environnement de travail . . . . .	57
4.3.1	Langage de programmation <i>Python</i> . . . . .	57
	Présentation du langage . . . . .	57
	Caractéristiques du langage et avantages . . . . .	57
4.3.2	Les principaux packages <i>Python</i> utilisés . . . . .	58
	Numpy . . . . .	58
	Scikit-learn . . . . .	58
	Pandas . . . . .	59

	Maths . . . . .	59
	time . . . . .	59
	random . . . . .	59
4.3.3	Outils et logiciels . . . . .	59
	Système d'exploitation (Ubuntu - GNU/Linux) . . . . .	59
	L'environnement de développement intégré (IDE) Pycharm	60
	Gestionnaire de versions <i>Git</i> . . . . .	60
	Editeur de texte : Sublime Text . . . . .	60
4.4	présentation de l'application . . . . .	61
4.4.1	Vue générale . . . . .	61
4.4.2	Description de l'application par visualisation d'un cas d'uti- lisation . . . . .	63
4.5	Expérimentations . . . . .	65
4.5.1	Jeux de données (base de test) . . . . .	65
	Présentation de MovieLens . . . . .	65
	Architecture du data-set . . . . .	65
	Exploitation du data-set . . . . .	66
4.5.2	Choix des métriques utilisées . . . . .	66
4.5.3	Évaluations . . . . .	67
	Filtrage classique . . . . .	67
	Filtrage classique basé sur KNN . . . . .	69
	Apport du traitement des valeurs manquantes sur la tech- nique précédente . . . . .	71
	Vers un filtrage ItemItem . . . . .	74
	Apport du clustering sur le filtrage collaboratif . . . . .	76
	Apport de l'optimisation sur le clustering . . . . .	78

	Comparaison avec l'état de l'art . . . . .	80
	Tableau comparatif . . . . .	81
4.6	Conclusion . . . . .	82



# Table des figures

1.1	La recommandation [22]	6
1.2	Filtrage cognitif [40]	7
2.1	Illustration d'un arbre de décision	22
2.2	Arborescence d'un clustering hiérarchique[1]	28
2.3	[57]	36
3.1	Notre approche de recommandation d'items	43
3.2	Clustering	47
3.3	Éparpillement initial des fourmis de manière aléatoire	48
3.4	Première mise à jour de la table de phéromones	49
3.5	Construction de nouvelles solutions	50
3.6	Évaporation de phéromone après chaque cycle	51
4.1	Schématisation de l'architecture du système	57
4.2	Logos des packages <i>Python</i> utilisés	58
4.3	Logos des outils utilisés	60
4.4	Choix du data-set	61

4.5	Choix de l'algorithme à tester - Fenêtre d'accueil . . . . .	62
4.6	Choix des paramètres . . . . .	63
4.7	visualisation des résultats . . . . .	64
4.8	Sauvegarde des résultats . . . . .	64
4.9	structuration du fichier des évaluations du data-set . . . . .	66
4.10	variation des résultats selon un filtrage classique en fonction du seuil choisi . . . . .	68
4.11	variation des résultats selon un filtrage classique basé sur KNN en fonction de paramètre k . . . . .	69
4.12	Variation des résultats selon un filtrage classique basé sur KNN en fonction du paramètre k avec un traitement des valeurs manquantes en utilisant "user average" . . . . .	72
4.13	Variation des résultats selon un filtrage classique basé sur KNN en fonction du paramètre k avec un traitement des valeurs manquantes en utilisant "item average" . . . . .	73
4.14	Variation des résultats selon un filtrage classique basé sur KNN en fonction de paramètre k avec un traitement des valeurs manquantes en utilisant "item average" . . . . .	74
4.15	UserUser vs ItemItem . . . . .	75
4.16	Évolution de la performance du système selon le nombre de clusters indiqué initialement (k-medoids) . . . . .	76
4.17	Évolution de la performance du système selon le nombre de clusters obtenus en sortie (k-medoids) . . . . .	77
4.18	Performances selon le paramètre <i>maxIter</i> de notre algorithme d'optimisation . . . . .	78
4.19	k-medoids VS k-medoids optimisé . . . . .	79
4.20	Technique de j. Bobadilla [30] . . . . .	81

# Introduction

L'essor qu'a connu le web a considérablement transformé l'expérience des utilisateurs d'internet. En effet, naviguer sur la toile revient à entrer en contact avec un grand nombre de prestataires de différents types de services. De nos jours l'utilisateur passe des heures sur le net où il bénéficie de plusieurs services électroniques dont ceux dédiés aux services de vente en ligne ou autres. La majorité de ces services sont basés sur le principe de prise en compte des avis des utilisateurs sur leurs expériences en vue de multiplier leurs taux de vente et donc leurs chiffres d'affaires.

Ces systèmes prennent en considération le problème de surcharge d'information causé par les interactions multiples des utilisateurs et le volume important de données généré. D'un autre côté, les utilisateurs souhaitent souvent avoir un feedback (retour d'information) d'utilisation de la part d'autres utilisateurs ayant des profils similaires avant d'effectuer n'importe quel achat.

Dans cette optique, la majorité de ces systèmes sont basés aujourd'hui sur des algorithmes de filtrage d'information et de manière plus précise du filtrage collaboratif (FC), dont l'objectif est de filtrer l'information et de ne suggérer aux utilisateurs que la liste d'items qui leurs sont les plus appropriées en prenant en compte l'avis des utilisateurs similaires.

Ainsi, l'utilisation du FC et l'amélioration des performances de cet algorithme intéresse toujours la communauté des chercheurs travaillant sur le domaine de la recommandation. Dans ce même contexte, nous nous intéressons dans le cadre de notre projet de fin d'études à la proposition d'une approche de recommandation d'items basée sur le FC amélioré. Notre approche se base sur l'utilisation des techniques de classification optimisées. Notre objectif étant de mieux constituer les groupes d'utilisateurs (i.e. optimiser la construction des clusters) avant de prédire l'intérêt qu'aurait un utilisateur sur un item donné.

Nous avons considéré l'algorithme des k-medoids pour notre processus de clustering. Pour l'optimisation, nous avons choisi la méta-heuristique des colonies de fourmis.

Afin d'évaluer les différents algorithmes, nous avons implémenté un prototype de systèmes de recommandation basé sur notre approche et avons effectué plusieurs expérimentations avec la base connue MovieLens. De plus, afin de comparer notre solution avec d'autres travaux de l'état de l'art, nous avons implémenté une autre approche déjà existante, basée sur l'optimisation du voisinage.

Notre mémoire sera organisé en deux parties, comme suit :

- **Une partie "État de l'art"**
  - **Chapitre 1** : Présente les systèmes de recommandation personnalisés.
  - **Chapitre 2** : Décrit les principales notions liées à la classification et l'optimisation.
- **Une partie "Contribution"**
  - **Chapitre 3** : Concerne la conception de notre approche, présentant les détails des différents algorithmes proposés..
  - **Chapitre 4** : Couvre les aspects techniques de l'implémentation de notre approche, ainsi que les évaluations effectués.

Finalement, une conclusion générale sera présentée incluant les résultats et les constatations obtenus, ainsi que des perspectives futurs.

Première partie

Etat de l'art

# Chapitre 1

## Recommandation personnalisée

### 1.1 Introduction

Avec une quantité de données en constante augmentation, le web devient plus que jamais une source d'information à grande échelle accessible à tous. Il devient ainsi compliqué de déterminer sur quoi porte l'intérêt de chaque internaute, car le filtrage de données aussi nombreuses que diverses est une tâche qui se montre de plus en plus complexe et par ailleurs, les informations qui pourraient s'avérer pertinentes pour un utilisateur  $\lambda$  demeurent difficiles à cibler.

Et c'est dans cette problématique qu'interviennent les systèmes de recommandation personnalisés. (Ils s'inscrivent donc parmi les systèmes de filtrage de l'information).

Indispensables, notamment en raison de l'essor du web, les systèmes de recommandation y sont aujourd'hui très présents et se montrent indispensables, voir incontournables.

Voici ce que nous allons présenter dans ce chapitre :

- Les principales notions liées aux systèmes de recommandation
- Les différentes techniques de recommandation
- La notion de profil utilisateur dans les systèmes de recommandation

- L'aspect sémantique dans la recommandation

## 1.2 Présentation des systèmes de recommandation

Nous vivons aujourd'hui l'ère du web, la masse d'individus connectés à internet est énorme, et ce en permanence. On y propose des services en tout genre, et l'on pourrait presque croire que le jour où nous n'aurons plus à quitter notre domicile pour faire quelque course qu'il soit est proche. Cependant une certaine problématique se pose. Les services et articles qu'on retrouve sur le web sont aussi nombreux que divers, et face à un choix aussi large il est très souvent difficile pour un utilisateur de s'y retrouver.

Certains acteurs se voient ainsi dans la nécessité de faire appel aux systèmes de recommandations personnalisées afin de faire face au problème. L'objectif est donc assez simple, il s'agit de produire auprès d'un « client » (visiteur d'un site, acheteur potentiel d'un produit, ...) une liste personnalisée de suggestions (produits, liens à cliquer, ...) en rapport avec ses préoccupations et attentes[41].

Aujourd'hui, la recommandation est adoptée par un grand nombre de sites-web et plateformes tels que Facebook, Amazon ou Netflix pour ne citer que ceux là, et pour certains d'entre eux l'enjeu est de taille. 30 % des ventes sur Amazon sont réalisées grâce à la recommandation [11], il va donc sans dire qu'un système de recommandation performant peut devenir un atout majeur dans la réussite de certains groupes.

## 1.3 Différentes approches pour la recommandation

Item est le terme utilisé pour désigner ce que le système recommande à l'utilisateur. Chaque item est considéré comme une unité élémentaire et insécable. Il peut s'agir de la musique à écouter, du livre à acheter, du film à visualiser, de l'article en ligne à lire etc. Un système de recommandation traite généralement un seul type d'items (CDs, films, livres, news) [25]. Les algorithmes et techniques de recommandation sont alors appliqués pour inférer des recommandations pertinentes relatives à ce type d'item [50]. Il s'agira alors de filtrer une certaine quantité d'informations afin d'en faire sortir les éléments pouvant s'avérer pertinents et intéressants pour l'utilisateur.



FIGURE 1.1 – La recommandation [22]

Pour ce faire, les systèmes de recommandation doivent donc connaître les goûts de chacun, ou du moins être en capacité de les déterminer. Ils se basent sur différentes caractéristiques de l'utilisateur afin d'être en mesure de cerner ses préférences.

Les approches utilisées diffèrent très souvent d'un système à un autre lorsqu'il s'agit de filtrer des informations, très souvent cela est du au type d'items manipulé. On retrouve d'ailleurs une multitude de combinaisons de techniques adoptée par chaque système de recommandation en fonction des problématiques auxquelles il fait face.

Les approches les plus utilisées sont : le filtrage collaboratif, le filtrage cognitif et le filtrage hybride. Nous allons donc les définir tout en mettant en valeur leurs principales caractéristiques ainsi que leurs différents avantages et inconvénients.

### 1.3.1 Filtrage cognitif (ou filtrage basé sur le contenu)

Les systèmes de recommandation basés sur le filtrage cognitif s'appuient essentiellement sur le contenu des items. On parle ainsi plus souvent de filtrage basé sur le contenu.

Un tel système recommande à l'utilisateur courant des items ayant une description similaire aux items qu'il a appréciés par le passé. Ce qui revient à identifier les caractéristiques communes aux items ayant reçu une évaluation favorable de la part de l'utilisateur courant. Ceci suppose qu'une description



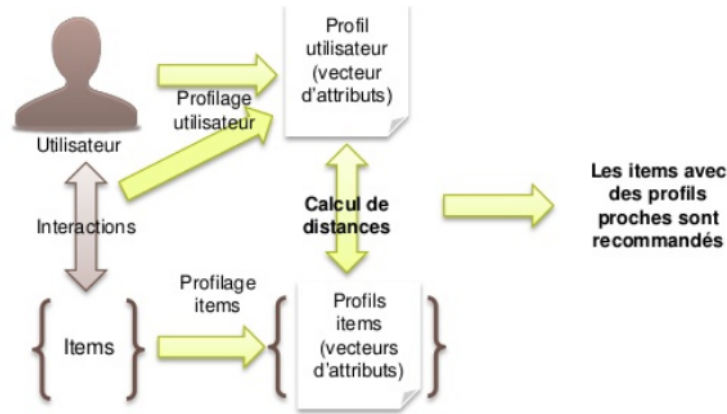


FIGURE 1.2 – Filtrage cognitif [40]

riche et variée est disponible sur les items à recommander [33][39][13]

Dans un système de recommandation basé sur le contenu, le profil de l'utilisateur est construit à partir de l'historique de ses évaluations, plus généralement à partir de l'analyse des usages [50].

Lors de l'arrivée d'un nouvel item le système compare sa représentation avec celle du profil utilisateur via un calcul de similarité, afin de déterminer les items susceptibles d'être intéressants pour cet utilisateur. Un profil doit donc être établi pour chaque item, tel que sa représentation est sous forme de collection d'enregistrements, représentant des caractéristiques.

Les auteurs de [34] découpent l'architecture d'un système de recommandation basé sur le contenu en trois composants : L'analyseur du contenu, l'apprentissage du profil utilisateur et le filtrage des items (ou recommandation) [50].

### 1. Analyse du contenu

Cette étape consiste à structurer les informations décrivant les collections d'items. Étant souvent présentés dans des formats textuel et non-structurés, il est donc nécessaire de les arranger de manière à les rendre exploitable par le système. Une analyse du contenu pourra ainsi être faite (mots clés, concepts etc ...)

### 2. L'apprentissage du profil utilisateur

Ce composant collecte les données issues de l'analyse des usages pour construire le profil utilisateur. Le profil utilisateur définit le modèle de préférences des utilisateurs pour le contenu des items à partir de leurs appréciations pour les items [50].

### 3. Filtrage

C'est la recommandation proprement dite. Une certaine similarité est calculé entre le profil de l'utilisateur précédemment établi et les descripteurs des items. Cette similarité permettra de définir les items pouvant s'avérer pertinents pour l'utilisateur en question.

## Avantages et inconvénients des systèmes basés sur le contenu

### • Avantages

- Le filtrage basé sur le contenu traite chaque utilisateur de façon indépendante [50]. Il n'est donc pas nécessaire de disposer d'une large communauté d'utilisateurs.
- Les items recommandés seront très similaires à ceux ayant déjà été bien évalués par l'utilisateur en question.
- L'ancienneté ou la popularité des items n'entre pas en compte, ce filtrage se base uniquement sur la thématique du contenu. Ainsi un nouvel item sera recommandé si son contenu est corrélé avec le profil de l'utilisateur.

### • Inconvénients

- Un système de recommandation basé sur le contenu n'est pas en mesure d'évaluer la qualité d'un item car les appréciations de la communauté d'utilisateurs ne sont pas pris en compte. Par exemple, dans un système de recommandation d'articles, si deux articles de presse traitent du même sujet, ils seront forcément décrits par les mêmes termes et auront par conséquent le même profil, le système n'aura aucun moyen de distinguer l'article de presse bien rédigé de celui mal rédigé [50].
- Un problème de sur-spécialisation survient très souvent. Effectivement, un utilisateur obtiendra toujours des recommandations d'items ayant des contenus proches. Même s'il existe des items peu corrélés avec son profil et qui sont susceptibles de l'intéresser, le système ne les proposera pas. Mais idéalement, l'utilisateur doit recevoir des recommandations pertinentes diversifiées et non homogènes [7].
- Il est nécessaire qu'un utilisateur évalue des items afin qu'un tel système puisse mettre en place son profil et être en mesure de lui proposer des items pertinents. Ce problème est connu dans la littérature sous le nom de démarrage à froid pour les utilisateurs (user cold start) [50].
- Une limite naturelle du filtrage basé sur le contenu est la nécessité de disposer d'une représentation variée et riche du contenu des items, ce qui n'est pas toujours le cas [50].

- Contrairement au filtrage collaboratif qui peut traiter tout type d'items, le filtrage basé sur le contenu ne peut traiter que les items disposant d'un contenu pouvant être analysé [50].

### 1.3.2 Filtrage collaboratif

Le filtrage collaboratif est basé sur le concept du « bouche à oreille ». Les opinions de toute la communauté d'utilisateurs sont ainsi pris en compte. Ceci part du principe qu'un utilisateur peut se voir amené à demander conseil ou à se renseigner auprès d'autres personnes à propos d'un produit ou d'un service afin de déterminer si celui-ci est réellement intéressant. Dans ce type de filtrage la seule information qui relie les utilisateurs aux items est leurs évaluations (ratings) sur ces derniers.

Cette approche peut être appliquée à tout type de documents textuels et multimédias car, contrairement au filtrage cognitif, elle ne se base pas sur le contenu des items, mais principalement sur l'exploitation des évaluations des utilisateurs, afin de recommander ces mêmes items à d'autres utilisateurs « similaires ».

Le système Tapestry [20] développé par Xerox PARC est le premier système à utiliser l'opinion de ses utilisateurs pour le filtrage de messages électroniques. En plus du contenu du message, son auteur et ses lecteurs, Tapestry stocke les avis des utilisateurs sous forme d'annotations (exemple "intéressant", "excellent"). L'utilisateur a la possibilité de définir son propre filtre en combinant des mots clés associés au contenu (par exemple "système de recommandation") avec des mots clés issus des annotations (annoté "excellent" par "Jean"). Le terme collaboratif, alors introduit pour la première fois, signifie que les utilisateurs collaborent entre eux pour s'entraider à filtrer leurs messages [50].

GroupLens, système expérimental de l'université du Minnesota, est l'un des plus célèbres et solides dans ce domaine. Il est semblable dans son esprit à Tapestry : les lecteurs sont appelés à noter les articles qu'ils lisent sur une échelle numérique de cinq niveaux. Le système trouve alors des corrélations entre les différents utilisateurs RI d'estimations et identifie des groupes d'utilisateurs dont les intérêts sont semblables, et ensuite il emploie ces estimations pour prédire l'intérêt que porteront les lecteurs à chaque article [18]

On distingue deux principales approches de filtrage collaboratif, le filtrage basé mémoire et le filtrage basé modèle :

### Filtrage collaboratif basé mémoire

Le filtrage collaboratif basé mémoire s'appuie sur les évaluations déjà faites par d'autres utilisateurs afin de pouvoir faire des prédictions. Il s'agit de connaître les évaluations faites par le voisinage d'un utilisateur afin de déterminer à quel degré celui-ci sera intéressé par un certain item.

Plus précisément, le filtrage collaboratif basé mémoire consiste à estimer les similarités entre les lignes et les colonnes de la matrice d'usage.

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

TABLE 1.1 – Matrice d'usage

Dans la matrice d'usage (Tableau 1.1), les colonnes sont représentées par l'ensemble des items et les lignes par l'ensemble des utilisateurs. L'intersection d'une colonne  $i$  avec une ligne  $j$  correspond à l'évaluation de l'utilisateur  $i$  sur l'item  $j$ . Dans la cas de la figure présentée, il s'agit d'une évaluation bornée par les valeurs 1 et 5. Une case vide signifie que l'utilisateur n'a jamais évalué cet item auparavant.

Pour ce type de filtrage, deux approches sont envisageables : basée sur les utilisateurs ou basée sur les items.

### Filtrage basé sur les utilisateurs (user-user ou user-based) :

Déterminer les voisins de l'utilisateur courant en calculant sa similarité avec les autres utilisateurs de la base [50]. Ainsi, il s'agit de regrouper les utilisateurs ayant des goûts en communs en comparant les lignes de la matrice d'usage. L'évaluation d'un utilisateur sur un item se prédit en s'appuyant sur les votes de ses voisins sur ce même item.

	$i_1$	$i_2$	$i_3$	$i_4$	$i_5$
$u_1$		2	4	3	1
$u_2$	3	1			
$u_3$	3		5		
$u_4$		4	2	3	1
$u_5$		4	3	1	4
$u_6$		3		1	4

TABLE 1.2 – Matrice d’usage

*Après filtrage et calcul de prédiction pour l’utilisateur  $U_6$  sur l’item  $I_3$*

	$i_1$	$i_2$	$i_3$	$i_4$	$i_5$
$u_1$		2	4	3	1
$u_2$	3	1			
$u_3$	3		5		
$u_4$		4	2	3	1
$u_5$		4	3	1	4
$u_6$		3	<b>3</b>	1	4

TABLE 1.3 – Matrice d’usage

Les items ayant été évalué par les mêmes individus ont été regroupés, et une évaluation sur l’un de ces items a été effectuée en fonction des notes du groupe d’utilisateurs ayant évalué cet item.

#### **Filtrage basé sur les items :**

Avec la même approche que le filtrage basé sur les utilisateurs, le filtrage «item-based » rassemble les items ayant été évalués plus ou moins par les mêmes utilisateurs. Une prédiction sera ainsi faite de la même façon.

	$i_1$	$i_2$	$i_3$	$i_4$	$i_5$
$u_1$		2	4	3	1
$u_2$	3	1			
$u_3$	3		5		
$u_4$		4	2	3	1
$u_5$		4	3	1	4
$u_6$		3		1	4

TABLE 1.4 – Matrice d’usage

Après filtrage et calcul de prédiction pour l'utilisateur  $U_6$  sur l'item  $I_3$

	$i_1$	$i_2$	$i_3$	$i_4$	$i_5$
$u_1$		2	4	3	1
$u_2$	3	1			
$u_3$	3		5		
$u_4$		4	2	3	1
$u_5$		4	3	1	4
$u_6$		3	3	1	4

TABLE 1.5 – Matrice d'usage

Le filtrage collaboratif basé mémoire s'effectue en deux étapes :

1. Calcul de similarité pour la détermination du voisinage
2. Prédiction

#### 1. Le calcul de similarité

Le calcul de similarité est une étape essentielle dans la processus de recommandation avec une approche basée sur le filtrage collaboratif. Celle-ci permettra de déterminer le voisinage de l'utilisateur actif afin de pouvoir effectuer par la suite des prédictions pertinentes.

Il existe une multitude de formules permettant de calculer la similarité. Cependant les plus répandues sont : Le coefficient de corrélation de Pearson et le Cosinus. Il a été cependant démontré que le coefficient de Pearson reste le plus utilisé et le plus performant en terme de pertinence de prédiction [46].

##### • Le coefficient de corrélation de Pearson

Il permet de calculer la distance entre deux utilisateurs  $u$  et  $v$ . Le coefficient de corrélation de Pearson mesure la liaison linéaire entre deux variables numériques en calculant le rapport entre leur covariance et le produit non nul de leur écart type. Il permet ainsi de mesurer la similarité en supprimant l'inconvénient de la variation des votes [50].

$$pearson(u, v) = \frac{\sum_{i \in I} (r_{u,i} - \bar{r}_u) \cdot (r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I} (r_{u,i} - \bar{r}_u)^2 \cdot \sum_{i \in I} (r_{v,i} - \bar{r}_v)^2}} \quad (1.1)$$

tel que :

- $r_{u,i}$  est l'estimation de l'utilisateur  $u$  sur l'item  $i$ .
- $r_{v,i}$  est l'estimation de l'utilisateur  $v$  sur l'item  $i$ .

- $\bar{r}_u$  est la moyenne de toutes les notes de l'utilisateur  $u$ .
- $\bar{r}_v$  est la moyenne de toutes les notes de l'utilisateur  $v$ .
- $I$  est l'ensemble des items
- **Le cosinus**

C'est une mesure de similarité entre deux objets  $a$  et  $b$ , très utilisée en recherche et en filtrage d'information [43], elle consiste à représenter les deux objets par deux vecteurs  $a$  et  $b$  et de mesurer le cosinus de l'angle formé par ces deux vecteurs [50].

$$\text{sim}(a, b) = \cos(\vec{x}_a, \vec{x}_b) = \frac{\vec{x}_a \cdot \vec{x}_b}{\|\vec{x}_a\| \|\vec{x}_b\|} \quad (1.2)$$

Cette méthode permet de calculer le poids de l'utilisateur  $u_j$  par rapport à l'utilisateur actif  $u_i$ , il est calculé comme un cosinus entre les vecteurs formés par les évaluations des utilisateurs, comme suit :

$$\text{Cosine}(u_i, u_j) = \frac{\sum_{k=10}^n r_{i,k} \cdot r_{j,k}}{\sqrt{\sum_{k=10}^n r_{i,k}^2 \cdot \sum_{k=10}^n r_{j,k}^2}} \quad (1.3)$$

## 2. La prédiction

Il s'agit de l'étape décisive dans le processus de recommandation. Elle consiste à prédire le degré d'intérêt de l'utilisateur actif sur un item en ce basant sur les évaluations déjà effectuées par son voisinage.

La façon la plus intuitive d'effectuer cette prédiction est de calculer la moyenne des votes du voisinage de l'utilisateur actif sur l'item en question. Cependant, la plupart du temps une pondération est associée aux utilisateurs qui composent le voisinage et ce par rapport à la proximité de chacun d'entre eux avec l'utilisateur actif. La formule définissant cette approche est la suivante [28] :

$$\text{Pred}(u_i, i_k) = \bar{r}(u_i) + \frac{\sum_{u_j \in V_i} \gamma(u_i, u_j) \times (r_{u_j, i_k} - \bar{r}(u_j))}{\sum_{u_j \in V_i} \gamma(u_i, u_j)} \quad (1.4)$$

tel que :

- $\gamma(u_i, u_j)$  : est la similarité entre un utilisateur  $u_i$  et son voisin  $u_j$ , tel que  $u_i \in V_i$

## Filtrage collaboratif basé modèle

Le filtrage collaboratif basé modèle s'appuie sur une base de données établie et constituée à partir des évaluations faites par la communauté d'utilisateurs.

Cet important historique servira à mettre en place un modèle probabiliste ou issu d'un apprentissage afin d'être en mesure d'émettre des prédictions.

D'un point de vue probabiliste, la tâche de prédiction d'une évaluation peut être vue comme le calcul de la valeur espérée d'une évaluation, étant donné ce que l'on sait d'un utilisateur [18].

Sur les nombreux modèles probabilistes existants, nous pouvons citer deux exemples qu'on retrouve très souvent : le modèle basé sur les clusters et le modèle basé sur les réseaux bayésiens

Le modèle à base de clusters repose sur le principe que certains groupes ou types d'utilisateurs capturent un ensemble commun de préférences et de goûts. Étant donné un tel groupe, les préférences concernant les différents items (sous la forme d'évaluations) sont indépendantes. Les paramètres du modèle, les probabilités d'appartenance à une classe  $\Pr(C = c)$ , et les probabilités conditionnelles des évaluations sachant la classe sont estimées à partir d'un ensemble d'exemples d'évaluations d'utilisateurs, appelé la base des évaluations [18].

Le modèle à base de réseaux bayésiens associe un nœud à chaque item. Les états pour chaque nœud correspondent aux valeurs d'évaluation possibles pour chaque item. On inclut également un état correspondant à l'absence d'évaluation pour les domaines où il n'y a pas d'interprétation naturelle pour les données manquantes. On peut alors appliquer un algorithme d'apprentissage de réseau bayésien sur la base d'exemples, où les évaluations manquantes sont associées à une valeur « pas d'évaluation ». Dans le réseau résultant de l'apprentissage, chaque item a un ensemble d'items « parents » qui sont les meilleurs prédicteurs de ses évaluations. Chaque table de probabilité conditionnelle est représentée par un arbre de décision qui code les probabilités conditionnelles pour ce nœud [18].

Plusieurs modèles probabilistes issus des domaines de recherche sur l'intelligence artificielle et sur l'apprentissage automatique ont été appliqués au filtrage collaboratif [50]. Certains ont défini le problème de recommandation comme étant un problème de classification [31]. Les réseaux Bayésien [31][19][55][17], les processus décisionnels de Markov [26] et les réseaux de neurones [42] ont aussi été utilisés pour faire de la recommandation [50].

Cependant ces techniques font face à certaines problématiques. En effet, elles sont sensibles à l'arrivée de nouveaux utilisateurs ou à l'insertion de nouveaux items. La phase d'apprentissage sera ainsi ré-effectuée au fur à mesure des mises à jours, et connaissant la forte complexité de ces algorithmes, elle peut s'avérer très coûteuse en temps et en ressources.

## Avantages et inconvénients du filtrage collaboratif

- **Avantages**



- Cette approche bénéficie des évaluations des autres. Un grand nombre d'utilisateurs dans la base améliore ainsi les performances du système de recommandation
  - L'utilisateur actif pourra découvrir de nouveaux genres d'items intéressants qui ne correspondent pas forcément à ce qu'il a l'habitude d'apprécier. Ceci est dû au fait que cette approche ne s'appuie guère sur la thématique du contenu des items. L'effet entonnoir est donc inhibé
  - Ce filtrage ne souffrira pas du problème de traitement de certains critères associés au contenu des items
- **Inconvénients**
- Les valeurs manquantes sont le principal handicap de ces systèmes. Ceci est dû au fait que certains items sont notés par très peu d'utilisateurs. De ce fait, la matrice des votes est une matrice creuse avec un taux de valeurs manquantes pouvant atteindre 95% voire 99 % du total des valeurs [50]. C'est le cas par exemple, du système de recommandation MovieLens 24 [37]
  - Démarrage à froid : toute nouvelle ressource ne peut être recommandée car elle n'a pas encore été évaluée. De même, pour un nouvel utilisateur, le système ne peut lui recommander une ressource car tant qu'il n'a pas effectué d'évaluations il ne pourra pas faire partie d'une communauté d'utilisateurs

## Filtrage hybride

Le filtrage hybride combine filtrage cognitif et filtrage collaboratif afin de tirer profit des avantages des deux approches et combler leurs manques. Sachant que le filtrage collaboratif a besoin d'avoir accès aux votes des utilisateurs, il ne peut être appliqué lorsque de nouveaux items se présentent car ceux-ci n'ont pas encore été évalués. Dans ce cas de figure, il serait plus judicieux de se tourner vers le filtrage basé sur le contenu et exploiter les informations décrivant l'item en question. Si l'on se positionne dans un exemple associé à une bibliothèque de livres numériques, un livre récemment ajouté à la plateforme n'a encore jamais été lu et n'a donc pas pu être évalué par les nombreux lecteurs inscrits sur le site. Le système de recommandation pourrait donc éventuellement exploiter la thématique de ce livre (Auteur, genre, édition etc.) afin de pouvoir le recommander.

Burke [14][15] décrit sept approches différentes pour l'hybridation, elles sont définies comme suit :

- **Pondération**  
Plusieurs techniques sont appliquées distinctement. Leurs résultats seront pondérés puis combinés afin d'obtenir une nouvelle recommandation.
- **Commutation**  
Une permutation des différentes techniques est effectuée par le système. Ce choix s'effectue en temps réel et dépend de plusieurs critères .
- **Mixte**  
Il s'agit de créer une liste de recommandation issue de plusieurs techniques de filtrage différentes.
- **Combinaison de caractéristiques**  
Il s'agit d'appliquer un seul algorithme de recommandation dans lequel sont prises en compte les approches des différentes techniques concernées.
- **Cascade**  
C'est une application successive de plusieurs techniques de recommandation. Après un premier traitement , les traitements suivants raffinent et enrichissent les résultats.
- **Augmentation de caactéristiques**  
La sortie obtenue après l'application d'une première technique de recommandation sera donnée en entrée à une seconde technique.
- **Méta niveau**  
Un autre moyen de combiner deux techniques de recommandation est d'utiliser le modèle généré par la première technique comme entrée de la seconde technique. Cette technique est différente de la technique d'augmentation de propriétés [50].

## 1.4 Profil utilisateur

Afin de pouvoir émettre des prédictions de qualité, le système de recommandation se base essentiellement sur les informations associées aux utilisateurs. Il est important que le profil de l'utilisateur soit décrit de façon formelle afin qu'il soit exploitable par le système.

Selon Thanh Trung, il existe actuellement trois méthodes principales pour acquérir ces informations : la méthode d'acquisition implicite, la méthode d'acquisition explicite et la méthode hybride [52].

### 1. Acquisition explicite

Dans cette approche, les caractéristiques du profil utilisateur sont acquises directement à travers des informations entrées par l'utilisateur lui-même. Le plus souvent, les systèmes de recommandation demandent aux utilisateurs d'évaluer les items en attribuant des notes ou des appréciations (likes, étoiles etc ..).

### 2. Acquisition implicite

Les caractéristiques implicites sont acquises en analysant le comportement de l'utilisateur. On parle souvent de profilage. Il s'agit ainsi de déduire implicitement des informations sur l'utilisateur en observant ses interactions avec l'application ou le site. Si par exemple, on se positionne dans un site de vidéos tel que Youtube, on pourrait déduire l'intérêt que porte l'utilisateur pour une vidéo en analysant le temps qu'il a passé à la regarder, ou en vérifiant si la vidéo a été complètement ou partiellement visualisée.

L'avantage de cette approche est qu'elle ne requiert pas beaucoup d'efforts des utilisateurs dans le processus de profilage. Cependant, dans plusieurs cas sa précision n'est pas aussi bonne que celle de la méthode d'acquisition explicite. Elle est plus difficile à interpréter et potentiellement « bruyante » [49].

### 3. Acquisition hybride

Cette technique consiste à combiner caractéristiques implicites et caractéristiques explicites afin d'établir et surtout de mettre à jour correctement le profil utilisateur. Le plus souvent, les utilisateurs sont amenés à entrer certaines informations explicites concernant leurs intérêts. Cependant les goûts des utilisateurs peuvent varier, et il serait embêtant pour un utilisateur de mettre à jour fréquemment ses goûts. C'est là que peut intervenir le profilage, car en suivant le comportement de l'utilisateur, le système de recommandation peut s'apercevoir indirectement des changements ou de l'évolution de ses goûts.

Il existe plusieurs techniques afin de représenter le profil d'un utilisateur de façon à mettre en évidence les informations qui le décrivent. Cela permettra au système de recommandation d'analyser ses goûts afin de lui proposer les recommandation adéquates.

## 1.5 Conclusion

Dans ce chapitre, nous avons présenté la recommandation et mis en avant son importance dans l'ère du big-Data et du web. Aujourd'hui les systèmes de recommandation sont donc très présents notamment dans les secteurs du service web et du e-commerce. Nous avons présenté les principales techniques utilisées par ces systèmes afin d'arriver à satisfaire une communauté d'utilisateurs de

plus en plus vaste et exigeante. Du filtrage cognitif au filtrage sémantique en passant par les techniques hybrides, les techniques sont nombreuses présentant chacune des avantages et des inconvénients.

La construction des profils utilisateurs est essentiel dans la recommandation notamment pour les techniques basées sur le contenu. Les modèles les plus connus sont le : le modèle vectorielle et le modèle basé sur les ontologies. L'aspect sémantique est très cité dans le domaine de la recommandation, il permet souvent de dresser un meilleur profil des utilisateurs et ainsi d'établir des communautés très corrélées. Ce qui permettra ainsi aux systèmes de recommandation d'améliorer la pertinence des items proposés.

## Chapitre 2

# Classification et optimisation

Avec l'essor du web, de plus en plus de données sont en circulation. C'est à ce niveau qu'intervient la science des données, une discipline qui se doit de manipuler cette masse de données le plus souvent en s'appuyant sur des techniques issues de l'intelligence artificielle. Parmi ces techniques, la classification et l'optimisation reviennent toujours. En effet, l'optimisation devient nécessaire voir incontournable lorsqu'il s'agit d'améliorer entre autres les temps de réponses ou la pertinence de résultats. Dans ce même contexte, la classification intervient très souvent. Elle incarne également de nombreux domaines de l'IA tel que le data-mining ou le machine learning.

Dans ce chapitre, nous tenterons de définir la classification, ses différents types et nuances ainsi que les méthodes de classification et notamment de clustering les plus connues dans la littérature. Puis, nous parlerons de l'optimisation en présentant quelques méta-heuristiques.

## 2.1 Qu'est ce que la classification ?

Depuis toujours, l'homme n'a cessé d'observer le monde et a toujours inconsciemment ressenti le besoin de le comprendre. Même si le terme classification reste relativement récent, le phénomène est cependant bien plus ancien. En observant la lune pour la première fois, les savants se demandèrent forcément de quoi il s'agissait, était-ce une planète, une étoile, un soleil ... la réponse n'était pas si évidente à l'époque. En découvrant les fossiles préhistoriques d'homini-dés, il était difficile de savoir s'il s'agissait d'ossements de singes ou d'humains. L'homme a donc toujours été amené à catégoriser les objets qui l'entourent en

observant leurs formes ou les caractéristiques qui leurs sont associées.

En informatique, la classification est un processus qui consiste à faire une analyse d'un grand ensemble de données associées à des catégories afin de générer un modèle (classifieur). Il sera ainsi possible de prédire la classe ou la catégorie d'une nouvelle donnée (instance) grâce à l'étude faite au préalable sur un ensemble de données déjà connu.

Un processus de classification s'effectue généralement en deux grandes phases bien distinctes : L'apprentissage et la prédiction (ou classification)

La phase d'apprentissage peut être considérée comme l'étape clé du processus. Un modèle sera mis en place suite à une analyse faite sur un ensemble de données d'apprentissage et tentant d'associer les caractéristiques de ces données à leurs catégories. Ces données peuvent être de différentes natures : objets, tuples etc.

Une fois l'apprentissage effectué, un classifieur est généré. Suite à l'arrivée d'un nouvel objet ou d'une nouvelle instance, le modèle sera en mesure d'émettre une prédiction sur la classe à laquelle appartiendrait l'instance en appliquant les règles obtenues lors de l'apprentissage.

Une classification de données peut se faire dans deux cadres différents : on parle de classification supervisée ou non-supervisée (clustering)

## 2.2 Classification supervisée

Dans ce cas, on parle souvent de classification tout court ou d'apprentissage supervisé. On parle de cadre supervisée car les classes des éléments à classer sont connues d'avance.

Si on se plaçait par rapport à une banque qui aurait comme problématique de savoir si un nouveau client serait intéressant ou plutôt à éviter, dans ce cas, les classes sont évidentes et sont connues : « bon client » ou « mauvais client ». Ainsi en se basant sur l'historique de la clientèle avec laquelle la banque a déjà traité, un modèle pourra être généré en effectuant un apprentissage sur la base de données après avoir étiqueté les clients déjà connus en leur attribuant comme étiquette « bon client » ou « mauvais client ». Il sera ainsi possible pour la banque de générer un classifieur plus ou moins efficace qui lui permettrait d'éviter les mauvais clients et ainsi d'empêcher d'éventuelles pertes.

Il existe plusieurs algorithmes de classification connus dans la littérature, nous ne les présenterons pas tous. Mais nous présenterons quelques-unes des

techniques les plus connues et les plus utilisées.

### 2.2.1 Les arbres de décisions

Un arbre de décision est une structure arborescente semblable à un organigramme, où chaque nœud interne (nœud non-feuille) indique un test sur un attribut, chaque branche représente un résultat du test et chaque nœud feuille (ou nœud terminal) détient une étiquette de classe. Le nœud le plus haut dans une arborescence est le nœud racine [27].

Contrairement à beaucoup de techniques de classification (régression logistique, SVM, etc.), les arbres de décision sont extrêmement intuitifs et fournissent une représentation graphique, parlante et facile à lire, d'un protocole de classification des individus [45].

Étant donnée l'ensemble d'apprentissage suivant :

Id	Age	Sexe	Salaire	Etat civique	Client intéressant
001	25	M	3200	Marié	Oui
002	35	F	1800	Célibataire	Oui
003	60	M	2300	Marié	Non
004	21	M	1800	Célibataire	Non
005	40	M	2400	Marié	Non
006	22	F	2000	Célibataire	Oui
007	55	F	2500	Marié	Oui

En effectuant un apprentissage sur cet ensemble, on obtiendrait un arbre de décision qu'on pourrait illustrer comme suit :

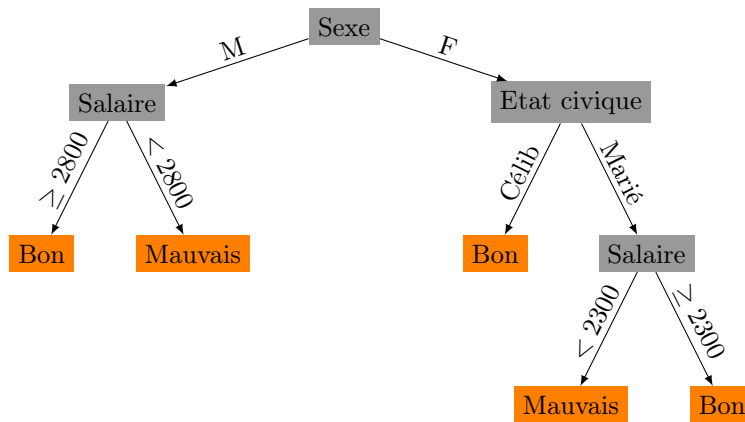


FIGURE 2.1 – Illustration d’un arbre de décision

Avec l’arrivée d’une nouvelle instance, ou plutôt d’un nouveau client dans notre cas, il sera possible de classifier cet individu et déterminer à partir de l’arbre de décision si celui-ci est un bon ou un mauvais client.

Exemple d’une nouvelle instance :

009	35	F	2200	M	?
-----	----	---	------	---	---

*En se basant sur l’arborescence obtenue, ce client sera considéré comme potentiellement mauvais, et ainsi il y a de forte chance que la banque rejette sa requête.*

009	35	F	2200	M	<b>Mauvais client</b>
-----	----	---	------	---	-----------------------

Bien évidemment, la construction de l’arbre suit certaines directives. Le choix de l’attribut à segmenter est basé sur certaines règles et ce afin d’obtenir le meilleur modèle possible. On retrouve plusieurs versions de cette algorithme, cependant les deux améliorations les plus connues de l’algorithme générique restent ID3 et C4.5 .

### 2.2.2 Les classifieurs bayésiens

Les classifieurs bayésiens sont des classifieurs statistiques. Ils peuvent prédire les probabilités d’appartenance à une classe telle que la probabilité qu’un tuple donné appartienne à une classe particulière[27].



La classification bayésienne est basée sur le théorème de Bayes. Ce dernier est un classique de la théorie des probabilités. Ce théorème est fondé sur les probabilités conditionnelles[12].

### Exemple

Dans une classe de 100 élèves de terminal, 60 sont encore mineurs et 40 d'entre eux ont réussi à l'examen du Baccalauréat. On peut ainsi noter cela de la façon suivante :

- $A = \text{"l'élève est mineur"}$

$$P(A) = \frac{60}{100} \quad (2.1)$$

- $B = \text{"l'élève a décroché le Baccalauréat"}$
- $A \cap B = \text{"l'élève est mineur et a réussi à décrocher le Baccalauréat"}$

$$P(A \cap B) = \frac{40}{100} \quad (2.2)$$

### Probabilité conditionnelle

$B \mid A = \text{"L'élève a décroché le baccalauréat sachant qu'il est mineur"}$

$$P(B|A) = \frac{40}{60} \quad (2.3)$$

On retient :

$$P(B|A) = \frac{P(A \cap B)}{P(A)} \quad (2.4)$$

Ou encore :

$$P(A \cap B) = P(A) \cdot P(B|A) \quad (2.5)$$

### La classification

Voici comment Han et al[27] explique le fonctionnement d'un classifieur bayésien simple :

- Soit  $D$  un ensemble d'apprentissage constitué de tuples et leurs étiquettes de classes associées. Chaque tuple est représenté par un vecteur d'attributs de dimension  $= n$ ,  $X = (x_1, x_2, \dots, x_n)$ , représentant  $n$  mesures effectuées sur le tuple à partir de  $n$  attributs, respectivement,  $A_1, A_2, \dots, A_n$ .

- Supposons qu'il y ait  $m$  classes,  $C_1, C_2, \dots, C_m$ . Étant donné un tuple  $X$ , le classificateur prédira que  $X$  appartient à la classe ayant la probabilité à posteriori la plus élevée, conditionnée par  $X$ . Autrement dit, le classificateur bayésien prédit que le tuple  $X$  appartient à la classe  $C_i$  si et seulement si :

$$P(C_i|X) > P(C_j|X) \text{ pour } 1 \leq j \leq m, j \neq i$$

Ainsi, nous maximisons  $P(C_i|X)$ . La classe  $C_i$  pour laquelle  $P(C_i|X)$  est maximisée s'appelle l'hypothèse maximum posteriori. Par le théorème de Bayes :

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)} \quad (2.6)$$

- Comme  $P(X)$  est constant pour toutes les classes, seul  $P(X|C_i)P(C_i)$  doit être maximisé. Si les probabilités à priori de la classe ne sont pas connues, alors on suppose généralement que les classes sont équiprobables, c'est-à-dire que  $P(C_1) = P(C_2) = \dots = P(C_m)$ , et nous maximisons  $P(X|C_i)$ . Sinon, nous maximisons  $P(X|C_i)P(C_i)$ . Notez que les probabilités a priori de la classe peuvent être estimées par  $P(C_i) = |C_{i,D}|/|D|$ , où  $|C_{i,D}|$  est le nombre de tuples d'apprentissage de classe  $C_i$  dans  $D$ .
- Sachant que les ensembles de données se composent de nombreux attributs, il serait extrêmement coûteux de calculer  $P(X|C_i)$ . Pour réduire le calcul dans l'évaluation de  $P(X|C_i)$ , l'hypothèse naïve de l'indépendance conditionnelle de classe sera prise en compte. Cela suppose que les valeurs des attributs sont conditionnellement indépendantes les uns des autres, étant donné l'étiquette de classe du tuple (c'est-à-dire qu'il n'y a pas de relation de dépendance entre les attributs). Ainsi,

$$P(X|C_i) = \prod_{k=1}^n P(x_k|C_i) = P(x_1|C_i) \times P(x_2|C_i) \times \dots \times P(x_n|C_i) \quad (2.7)$$

Nous pouvons facilement estimer les probabilités  $P(x_1|C_i), P(x_2|C_i), \dots, P(x_n|C_i)$  des tuples d'apprentissage. Rappelons que  $x_k$  se réfère à la valeur de l'attribut  $A_k$  pour le tuple  $X$ . Pour chaque attribut, nous regardons si l'attribut est catégorique ou continu. Par exemple, pour calculer  $P(X|C_i)$ , nous considérons ce qui suit :

- Si  $A_k$  est catégorique, alors  $P(x_k|C_i)$  est le nombre de tuples de classe  $C_i$  dans  $D$  ayant la valeur  $x_k$  pour  $A_k$ , divisée par  $|C_{i,D}|$ , le nombre de tuples de classe  $C_i$  en  $D$
- Si  $A_k$  est une valeur continue, alors nous devons faire un peu plus de travail, mais le calcul est assez simple. Un attribut à valeur continue est généralement supposé avoir une distribution gaussienne avec une moyenne  $\mu$  et un écart-type  $\sigma$ , défini par :

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (2.8)$$

Alors,

$$P(x_k|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i}) \quad (2.9)$$

Nous devons calculer  $\mu_{C_i}$  et  $\sigma_{C_i}$ , qui sont la moyenne et l'écart-type, respectivement, des valeurs de l'attribut  $A_k$  pour les tuples d'apprentissage de la classe  $C_i$ . Nous branchons ensuite ces deux valeurs dans l'équation. (2.8), avec  $x_k$ , pour estimer  $P(x_k|C_i)$ .

- Pour prédire la classe de X,  $P(X|C_i)P(C_i)$  est évalué pour chaque classe  $C_i$ . Le classificateur prédit que la classe du tuple X est la classe  $C_i$  si et seulement si :

$$P(X|C_i)P(C_i) > P(X|C_j)P(C_j) \quad \text{pour } 1 \leq j \leq m, j \neq i. \quad (2.10)$$

En d'autres termes, la classe prédite est la classe  $C_i$  pour laquelle la valeur de  $P(X|C_i)P(C_i)$  est la plus grande.

### 2.2.3 Entre autres

#### SVM

Les machines à vecteurs de support ou séparateurs à vaste marge (en anglais support vector machine, SVM) sont un outil mathématique très performant pour les problèmes de classification ainsi que pour les problèmes de régression.

Les SVM ont été inventés par Vladimir N. Vapnik et Alexey Ya. Chervonenkis en 1963. Depuis lors, les systèmes ont été utilisés dans le texte, l'hypertexte et pour la classification d'images. Les SVM peuvent travailler sur des caractères manuscrits. Ils furent également utilisés dans les laboratoires de biologie pour effectuer des tâches comme le tri des protéines[6].

#### Classifieur KNN

La méthode des k plus proches voisins, aussi simpliste qu'elle soit, reste l'une des techniques les plus utilisées et les plus performantes pour les problèmes de classification.

Étant donné un ensemble d'apprentissage, la classe d'une nouvelle entrée sera équivalente à la classe majoritaire parmi les classes associées aux k voisins

de cette même entrée. Le voisinage sera déterminé en se basant sur une mesure de distance ou sur une fonction de corrélation.

*Une multitude d'algorithmes et d'approches différentes furent proposés pour résoudre les problèmes de classification ou de régression. Il est cependant important de souligner que le choix d'une technique de classification dépend du problème donné. Il est même parfois impératif de passer par des expérimentations*

## 2.3 Clustering

Le clustering est une technique d'apprentissage automatique qui implique le regroupement de points de données. Étant donné un ensemble de points de données, nous pouvons utiliser un algorithme de classification pour classer chaque point de données dans un groupe spécifique. En théorie, les points de données appartenant au même groupe doivent avoir des propriétés et / ou des caractéristiques similaires, tandis que les points de données situés dans des groupes différents doivent avoir des propriétés et / ou des caractéristiques très différentes. Le regroupement est une méthode d'apprentissage non supervisé et est une technique courante pour l'analyse de données statistiques utilisée dans de nombreux domaines[47].

Il est possible de juger la qualité d'un clustering grâce à la distance séparant les éléments appartenant à un même cluster ainsi que la distance séparant les différents clusters générés par le clustering, on parle souvent de distance intra-cluster et de distance inter-cluster, dont les valeurs doivent respectivement être minimisées et maximisées.

La notion de cluster dépend de l'application et est généralement faiblement définie [38]. L'objectif de la tâche d'analyse de cluster affecte également la définition. Selon l'application, les clusters ont des formes et des tailles différentes

Voici quelques définitions collectées à partir de la littérature [29, 8] :

- "Un cluster est un ensemble d'entités qui se ressemblent, et les entités de différents clusters ne se ressemblent pas."
- "Un cluster est une agrégation de points dans l'espace, de sorte que la distance entre deux points dans le cluster est inférieure à la distance entre un point quelconque du cluster et un point qui n'en fait pas partie."

- "Les clusters peuvent être décrits comme des régions connectées d'un espace multidimensionnel contenant une densité relativement élevée de points, séparés d'autres régions similaires par une région contenant une densité de points relativement faible."

### 2.3.1 Les différents types de clustering

Il existe de nombreux algorithmes de clustering dans la littérature. Il est difficile de fournir une catégorisation nette des méthodes de regroupement, car ces catégories peuvent se chevaucher de sorte qu'une méthode peut avoir des caractéristiques de plusieurs autres catégories. Néanmoins, il est utile de présenter une image relativement organisée des méthodes de clustering [47]. D'après [27], les principales méthodes de classification fondamentales peuvent être classées selon les catégories suivantes :

#### Clustering par partitionnement(partionning-based clustering)

Les méthodes de clustering basées sur le partitionnement sont des méthodes flexibles basées sur le déplacement itératif des points entre les clusters[56]. Étant donné un nombre  $k$  de clusters, un premier partitionnement sera fait initialement. Celui-ci sera par la suite amélioré de façon progressive en remplaçant de manière itérative les différents points de l'ensemble de données d'un cluster à un autre. La majorité de ces méthodes sont basées sur une mesure de distance qui permet d'estimer la ressemblance entre deux points donnés. Une façon d'évaluer un partitionnement est de juger si les points appartenant à des mêmes clusters sont assez proches, et d'autre part, si les points appartenant à des clusters différents sont suffisamment éloignés ou différents[27].

#### Clustering hiérarchique(hierarchical clustering)

Dans le clustering hiérarchique, les données ne sont pas partitionnées dans un cluster particulier en une seule étape. Au lieu de cela, une série de partitionnements a lieu, pouvant aller d'un seul cluster contenant tous les objets à  $n$  clusters contenant chacun un seul objet. La classification hiérarchique est subdivisée en méthodes agglomératives, qui procèdent par une série de fusions des  $n$  objets en groupes, et des méthodes par divisions, qui séparent  $n$  objets successivement en groupes plus fins.

#### L'approche agglomérative (Agglomerative Hierarchical Clustering)

Étant donnée une population de  $n$  individus, on considère  $n$  clusters tel chaque

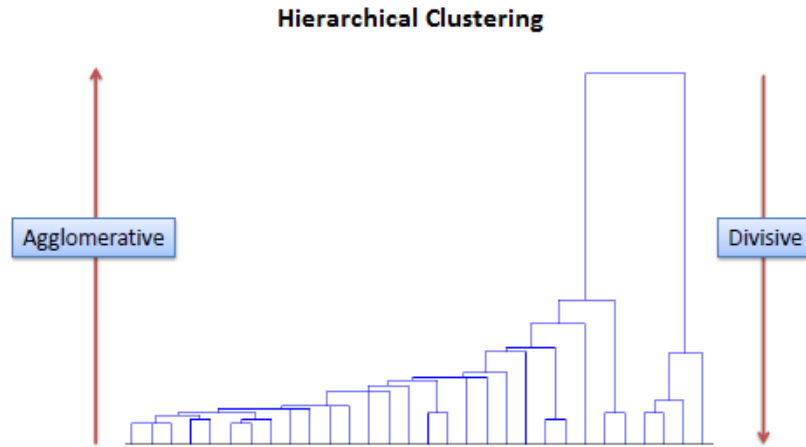


FIGURE 2.2 – Arborescence d'un clustering hiérarchique[1]

cluster contient un seul individu. De façon successive, les clusters les plus proches sont fusionnés deux à deux suivant une certaine mesure de distance ou de dissimilarité. Ce même processus est ainsi réitéré jusqu'à arriver à obtenir un seul et unique cluster contenant tout les points de notre espace.

Il sera donc possible d'illustrer l'ensemble du processus par une arborescence, qui nous permettra d'élaguer l'arbre au niveau souhaité afin de dégager les clusters désirés.

**L'approche par divisionS(divisive method)** A l'inverse de la technique précédente, cette approche consiste à partir d'un seul cluster et d'effectuer des divisions successives. On parle d'approche descendante.

### Clustering basé sur la densité (density-based method)

Le clustering basée sur la densité détecte les zones où les points sont concentrés et celles où les points sont séparés par des zones vides ou clairsemées. Les points ne faisant pas partie d'un agrégat sont classés comme bruit[2]. De nombreuses méthodes de clustering basées sur la densité furent proposées. Leur idée générale est de continuer à faire croître un cluster donné tant que la densité (nombre d'objets ou de points de données) dans le «voisinage» dépasse un certain seuil[27].

## L'approche basée sur la grille (grid-based method)

Un tel algorithme définit un ensemble de cellules-grilles, puis assigne chaque point à la cellule de grille appropriée et calcule la densité de chaque cellule. Les cellules dont la densité est inférieure à un certain seuil sont éliminées[48]. L'utilisation de grilles est souvent une approche efficace pour de nombreux problèmes d'exploration de données spatiales, y compris le clustering. Par conséquent, les méthodes basées sur la grille peuvent être intégrées à d'autres méthodes de clustering telles que les méthodes basées sur la densité et les méthodes hiérarchiques[27].

### 2.3.2 Quelques algorithmes de clustering

#### K-means

Sans doute l'un des algorithmes les plus réputés et les plus utilisés pour le clustering, k-means reste une référence dans ce contexte.

L'algorithme prend en entrée l'ensemble des données et le nombre de clusters  $k$ . Initialement, des points seront choisis aléatoirement ou à partir de l'ensemble des données comme centroïdes. un raffinement sera effectué de manière itérative de telle sorte à retourner les différents clusters définis ainsi que les coordonnées des centroïdes.

La qualité d'un cluster  $C_i$  peut être évaluée par la somme des erreurs quadratiques entre tous les objets de  $C_i$  et le centroïde  $c_i$ , telle que [27] :

$$E = \sum_{i=1}^k \sum_{p \in C_i} dist(p, c_i)^2 \quad (2.11)$$

**Algorithme : K-Means :** les centres de chaque cluster son représenté par la moyennes des valeurs des points qu'ils contiennent **Entrées :**

1.  $k$  : le nombre de clusters.
2. L'ensemble des données

**Sorties :**

1. centroïdes
2. Clusters de sortie

**méthode :**

- Choisir  $k$  points à partir de l'ensemble des données comme centroïdes initiaux
- **Répéter**
  - Pour chaque point, calculer la similarité entre celui-ci et les différents centroïdes et ainsi assigner le point au cluster approprié en tenant compte de la similarité la plus élevé.
  - Mettre à jour les centroïdes des différents clusters en calculant la moyenne des valeurs des points contenus dans chacun des clusters.

**Jusqu'à :** Aucun changement

L'algorithme k-means ne garantie en aucun cas le résultat optimal, mais a l'avantage de converger rapidement vers un optimum qui reste dépendant des centroïdes choisis initialement. K-means reste cependant décrits comme étant très sensible aux données aberrantes (outliers).

## K-Medoids

Souvent présenté comme une amélioration du k-means clustering, k-medoids reste cependant très similaire à ce dernier en ayant l'avantage d'être moins sensible aux données aberrantes. Les centroïdes, sont représentés par des points de l'ensemble des données, on parle généralement de medoids.

L'idée de base de cet algorithme est de déterminer d'abord les  $K$  objets représentatifs appelés medoids. Après avoir trouvé l'ensemble des medoids, chaque objet de l'ensemble de données est assigné au medoid le plus proche. C'est-à-dire que l'objet  $i$  est placé dans le cluster  $v_i$ , lorsque le  $mv_i$  medoid est plus proche que tout autre  $m_w$  medoid[16].

Proposé par Kaufman et Rousseeuw [15], l'algorithme PAM est la représentation la plus populaire des k-medoids, voici comment celui-ci procède[27] :



**Algorithmme : K-Medoids : PAM Entrées :**

1.  $k$  : le nombre de clusters.
2. L'ensemble des données

**Sorties :**

1. centroïdes
2. Clusters de sortie

**méthode :**

- Choisir  $k$  points à partir de l'ensemble des données comme centroïdes initiaux
- **Répéter**
  - Pour chaque point, calculer la similarité entre celui-ci et les différents centroïdes et ainsi assigner le point au cluster approprié en tenant compte de la similarité la plus élevée.
  - Sélectionner aléatoirement un point non-représentatif  $o_{random}$
  - calculer le coût total,  $S$ , de l'échange du point représentatif (medoïd)  $o_j$  avec  $o_{random}$
  - si  $S < 0$  alors permuter  $o_j$  avec  $o_{random}$  pour former le nouvel ensemble des  $k$  points représentatifs (medoïds)

**Jusqu'à :** Aucun changement

**Exemple :** Déroulement d'un exemple

On considère  $n$  points dans un espace bidimensionnel, tel que l'ensemble des points est défini comme suit :

point	$x$	$y$
1	2	4
2	3	8
3	5	7
4	3	1
5	1	3
6	6	1
7	4	7
8	6	5
9	5	1
10	2	3

TABLE 2.1 – Ensemble de points

Étant donné  $p_1$  et  $p_6$  les deux medoïds pris initialement, deux clusters seront

formés en affectant chaque point au cluster approprié en calculant la distance entre chaque point et les medoïds des différents clusters, pour notre exemple on utilisera la distance de Manhattan, cependant d'autres mesures de distance peuvent être choisies selon le type de problème.

L'ensemble des points			Distances	
point p	$x_p$	$y_p$	$m_1 = (2, 4)$	$m_2 = (6, 1)$
2	3	8	<b>5</b>	10
3	5	7	<b>6</b>	7
4	3	1	4	<b>3</b>
5	1	3	<b>2</b>	7
7	4	7	<b>5</b>	8
8	6	5	5	<b>4</b>
9	5	1	6	<b>1</b>
10	2	3	<b>1</b>	6

TABLE 2.2 – Illustration de l'algorithme sur l'ensemble des points

Ainsi, on obtient les deux clusters  $C_1 = 1, 2, 3, 5, 7, 10$  et  $C_2 = 6, 4, 8, 9$ . Le coût total de ce clustering est défini par la somme des distances entre chaque point de l'ensemble de données et le centre du cluster auquel il appartient.  $S = (5 + 6 + 2 + 5 + 1) + (3 + 5 + 1) = 28$

Dans un second temps, un point sera pris aléatoirement à partir de l'ensemble des données (le point 4 par exemple) et sera supposé comme un medoid éventuel. On aura ainsi :

L'ensemble des points			Distances	
point p	$x_p$	$y_p$	$newM_1 = (3, 1)$	$m_2 = (6, 1)$
1	2	4	<b>4</b>	7
2	3	8	<b>7</b>	10
3	5	7	8	<b>7</b>
5	1	3	<b>4</b>	7
7	4	7	<b>7</b>	8
8	6	5	7	<b>4</b>
9	5	1	2	<b>1</b>
10	2	3	<b>3</b>	6

TABLE 2.3 – Illustration de l'algorithme sur l'ensemble des points(suite)

Le nouveau coût est de :  $S = (4 + 7 + 4 + 7 + 3) + (7 + 4 + 1) = 37$

Comme la différence entre le nouveau coût et l'ancien est positive  $NewC - OldC > 0$  alors il n'y a pas d'amélioration et donc l'algorithme prend fin et

retourne les clusters :  $C_1 = 1, 2, 3, 5, 7, 10$  et  $C_2 = 6, 4, 8, 9$  et les points 1 et 6 comme medoids des deux clusters.

## DBSCAN

DBSCAN (Clustering spatial basé sur la densité et application avec bruit), est un algorithme de clustering basé sur la densité, introduit par M.Ester and all [36]. L'algorithme DBSCAN nécessite 2 principaux paramètres [44] :

- **eps** : la distance minimale entre deux points. Cela signifie que si la distance entre deux points est inférieure ou égale à cette valeur (eps), ces points sont considérés comme des voisins.
- **minPoints** : le nombre minimum de points pour former une région dense. Par exemple, si nous définissons le paramètre minPoints sur 5, nous avons besoin d'au moins 5 points pour former une région dense.

Le choix des paramètres pose naturellement problème. En effet, avec une distance minimale trop petite, il arrive que de nombreux points ne soient associés à aucun cluster ce qui fait d'eux des *outliers*. Cependant une valeur trop élevée fera en sorte que de nombreux points, même très différents se retrouvent dans des mêmes clusters.

L'algorithme parcourt tout les points de l'ensemble de données, en commençant par un point aléatoire et lançant par là la construction d'un cluster à partir de ce même point. Tout autre point avec une distance le séparant du point choisi inférieure à *eps* sera inclut dans le cluster. La même procédure sera effectuée avec les points ajoutés au cluster et ce jusqu'à ce qu'il n'y ait plus de point à ajouter dans ce dernier. À ce niveau, si la taille du cluster est supérieure à *minPoints* alors le cluster sera gardé, sinon les points qu'il contient seront considérés comme des "*outliers*".

### 2.3.3 Le clustering dans les systèmes de recommandation

Aujourd'hui une tâche de clustering peut s'avérer nécessaire pour de nombreux problèmes informatiques ou autres. En effet, un partitionnement de données est souvent indispensable dans le but d'effectuer un traitement quelconque ou de dégager quelque information. Le clustering est particulièrement très utilisé en data-mining.

Le clustering intervient aussi très souvent dans la recommandation, en effet de nombreuses approches basées sur le clustering furent proposées. Dans un de leurs travaux, Lyle and all [51] affirment que des prédictions beaucoup plus précises peuvent être faites en regroupant les gens en groupes avec des films similaires et en regroupant les films en groupes qui ont tendance à être aimés

par les mêmes personnes. Si vous et moi avons aimé les mêmes films, alors je vais probablement profiter d'autres films que vous aimez.

Dans un contexte de filtrage collaboratif basé mémoire, Manh Cuong Pham and all [35] partitionnent les utilisateurs en communauté en se basant sur les informations sociales.

Kohrs [32] ont utilisé un algorithme de classification hiérarchique pour regrouper indépendamment les utilisateurs et les éléments dans deux hiérarchies de clusters.

## 2.4 Méta-heuristiques pour l'optimisation

Apparues au début des années 1980 [24], les méta-heuristiques s'affirment comme des méthodes génériques pour la résolution de problèmes combinatoires NPdifficiles. La résolution de ces problèmes nécessite l'examen d'un très grand nombre (exponentiel) de combinaisons[23]. Ainsi, plus la taille du problème augmente plus les chances de se heurter à une explosion combinatoire sont grandes lorsque qu'on a recours à des solutions exactes. Les problèmes d'identification, l'apprentissage supervisé de réseaux de neurones ou encore la recherche du plus court chemin sont, par exemple, des problèmes d'optimisation[57]. Dans ce contexte, c'est souvent le temps de calcul qui pose problème, ou celui-ci doit être optimisé au dépend de la qualité de la solution ou celle-ci n'est pas toujours la solution optimale (optimum global) au problème traité.

Aujourd'hui, l'intelligence en essaim est en train de remplacer les méta-heuristiques traditionnelles. En effet, celle-ci prend en compte une intelligence collective inspirée de la nature. C'est ainsi qu'elle naquit, et elle trouve aujourd'hui des applications dans le domaine de la simulation et au-delà, par exemple en robotique collective ou pour les réseaux. La recherche sur le comportement collectif des insectes sociaux a fourni aux informaticiens des méthodes puissantes pour la conception d'algorithmes d'optimisation et de contrôle distribué[21], Dorigo fut l'un des premiers à proposer une telle approche inspirée du comportement des fourmis.

### 2.4.1 Les algorithmes génétiques

Les algorithmes génétiques utilisent la théorie de Darwin sur l'évolution des espèces. Ils fournissent des solutions aux problèmes n'ayant pas de solutions calculables en temps raisonnable de façon analytique ou algorithmique[9]. Cette approche repose sur trois principes[3] : le principe de variation, le principe d'adaptation et le principe d'hérédité .

- **Le principe de variation** : Chaque individu au sein d'une population est unique. Ces différences, plus ou moins importantes, vont être décisives dans le processus de sélection.
- **Le principe d'adaptation** : Les individus les plus adaptés à leur environnement atteignent plus facilement l'âge adulte. Ceux ayant une meilleure capacité de survie pourront donc se reproduire davantage.
- **Le principe d'hérédité** : Les caractéristiques des individus doivent être héréditaires pour pouvoir être transmises à leur descendance. Ce mécanisme permettra de faire évoluer l'espèce pour partager les caractéristiques avantageuses à sa survie.

Avant le lancement de l'algorithme, le codage des individus, qui représentent un ensemble de solutions, est une étape primordiale. La façon la plus courante pour codifier les individus est de convertir chaque valeur en un vecteur de bits.

L'algorithme s'appuie sur trois opérations de base :

#### 1. Sélection

Selon une certaine fonction objectif ou une mesure de coût/qualité, une partie des individus de la population sera choisie. Tout comme la nature l'impose, les plus forts ont plus de chance de survivre alors que les plus faibles finissent souvent par disparaître. Celle-ci s'inspire donc de la sélection naturelle.

#### 2. Croisement

Suite à la sélection des meilleurs individus, un croisement (reproduction) est effectué entre ces derniers, donnant naissance à de nouveaux individus tels que chacun dispose de ses propres caractéristiques. L'objectif de cette opération est de permettre la recombinaison des informations présentes dans le patrimoine génétique de la population[22].

#### 3. Mutation

Cet opérateur consiste à changer la valeur allélique d'un gène avec une probabilité  $pm$  très faible, généralement comprise entre 0.01 et 0.001[9]. Étant donnée un ensemble de vecteurs de bits représentant nos individus, une mutation consisterait à inverser la valeur d'un des bits. Ceci arrive très rarement étant donnée la probabilité infime de l'événement.

Les algorithmes génétiques peuvent être adaptés et appliqués à de nombreux problèmes. Même s'ils se montrent souvent peu efficaces [10], ils apportent cependant assez rapidement une solution acceptable.

### 2.4.2 Optimisation par colonie de fourmis

Les fourmis adoptent un comportement collectif où chacune d'entre elles se met à la disposition de la communauté. En effet, celles-ci, bien qu'ayant

individuellement des capacités cognitives limitées, sont capables collectivement de trouver le chemin le plus court entre une source de nourriture et leur nid. Des biologistes ont d'ailleurs observé, dans une série d'expériences menées à partir de 1989, qu'une colonie de fourmis ayant le choix entre deux chemins d'inégale longueur menant à une source de nourriture avait tendance à utiliser le chemin le plus court[5] comme le montre le schémas ci-dessous.

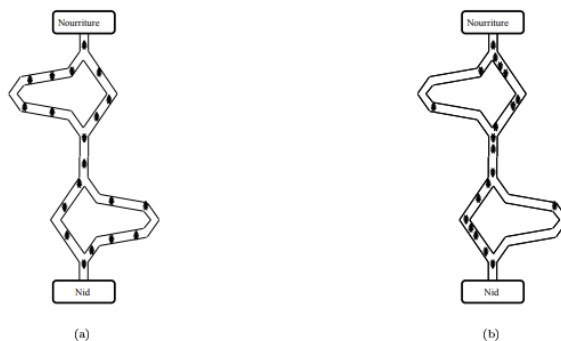


FIGURE 2.3 – [57]

Au début du processus de recherche de ressource, les fourmis s'éparpillent le plus souvent de manière aléatoire. Très souvent cela se développe dans le cadre d'un système multi agents où chaque agent fait référence à une fourmi. Il a surtout été observé que les fourmis ne sont pas réellement dotés de sens mais sont cependant réactives à une substance chimique qu'on appelle "phéromone", celle-ci est d'ailleurs secrétée par les fourmis afin d'être en mesure de retrouver le chemin de leur nid. Les colonies de fourmis artificielles s'inspirent de cette caractéristique afin de converger vers une bonne solution dans un temps relativement court.

Cette méta-heuristique est relativement récente. Elle a été introduite en 1991 par Colomi, Dorigo et Maniezzo pour résoudre le problème du Voyageur de commerce. Elle s'est popularisée, puis a été l'objet d'améliorations dès 1995 et a été appliquée avec succès à d'autres problèmes d'optimisation combinatoire dès 1994[10].

L'algorithme AS est formalisé de façon à être adapté au problème du voyageur de commerce qui consiste à trouver un circuit hamiltonien de longueur minimale sur un graphe. Il s'agit d'un problème NP-Complet.

A tout instant  $t$ , chaque fourmi choisit une ville de destination selon un choix défini. Toutes les fourmis se placent à l'instant  $t + 1$  dans une ville de leurs choix. On appelle une itération de l'algorithme AS, l'ensemble de déplacements de l'ensemble de la colonie entre l'instant  $t$  et l'instant  $t + 1$ . Ainsi après  $n$

itérations, l'ensemble de la colonie aura effectué un circuit hamiltonien sur le graphe. De cette manière toutes les fourmis commenceront et finiront leur tour au même temps.

Se trouvant déjà sur une ville à l'instant  $t$ , chaque fourmi décidera de sa prochaine destination selon un certain choix qui s'offre à elle dépendamment de la quantité de phéromone qui a été déposée sur les transitions. Cependant, il existe une certaine mémoire au sein de chaque fourmis qui permet à ces dernières de ne pas repasser par la même ville.

D'après Dorigo, le choix des transitions devra s'effectuer comme suit : Une fourmi  $k$  placée sur la ville  $i$  à l'instant  $t$  va choisir sa ville  $j$  de destination en fonction de la visibilité  $\eta_{ij}$  de cette ville et de la quantité de phéromones  $\tau_{ij}(t)$  déposée sur l'arc reliant ces deux villes. Ce choix sera réalisé de manière aléatoire, avec une probabilité de choisir la ville  $j$  donnée :

$$P_{ij}^k(t) = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha \times \eta_{ij}^\beta}{\sum_{l \in N_i^k(t)} [\eta_{il}(t)]^\alpha \times \eta_{il}^\beta} & \text{si } x \geq 0 \\ 0 & \text{sinon} \end{cases} \quad (2.12)$$

tel que :

- $k$  : indice de la fourmi
- $i$  : indice de la ville
- $N_i^k$  : l'ensemble des villes que la fourmi  $k$ , placée sur la ville  $i$ , n'a pas encore visité à l'instant  $t$  dans le cycle courant
- $\alpha$  et  $\beta$  : deux paramètres qui contrôlent l'importance relative entre phéromones et visibilité

A chaque fin de cycle, une mise à jour des phéromones est effectuée selon l'opération suivante :

$$\tau_{ij}(t+n) = \rho \times \tau_{ij}(t) + \Delta_{\tau_{ij}}(t) \quad (2.13)$$

tel que :

- $\rho \in [0, 1[$  : paramètre empirique qui définira la vitesse d'évaporation des phéromones sur les arcs entre l'instant  $t$  et l'instant  $t+n$ .
- $\Delta_{\tau_{ij}}(t)$  : représente la quantité de phéromone déposée par chaque fourmi sur l'arc  $(i, j)$ .

Le choix de  $\rho$  est important, en effet si  $\rho$  se rapproche trop de 1, on observe un effet de stagnation des phéromones sur les arcs, ce qui implique des inconvénients tel que le fait de voir les mauvaises solutions persister. De même, choisir  $\rho \approx 0$  implique une évaporation trop rapide des phéromones, donc amène la fourmi à un choix dépendant uniquement de la visibilité des nœuds.

Appelons  $T_k(t) = u(k_1, \dots, u_q)$  le tour réalisé par la  $k$ -ème fourmi dans l'intervalle de temps  $[t, t+, n]$ , et  $L_k(t)$  sa longueur.  $T_k(t)$  (et donc  $L_k(t)$ ) s'obtient en analysant la mémoire de la fourmi. Soit  $\Delta_{\tau_{ij}^k}(t)$ , la quantité de phéromone déposée par cette fourmi sur l'arc  $(i, j)$  dans ce même intervalle de temps. On le définit ainsi :

$$\Delta_{\tau_{ij}^k}(t) = \begin{cases} Q/L_k(t) & \text{si } (u \in T_k(t) \wedge u = (i, j)) \\ 0 & \text{sinon} \end{cases} \quad (2.14)$$

où  $Q$  est une constante. On voit bien ici que les phéromones sont régulées en fonction de la qualité de la solution obtenue car plus  $L_k(t)$  est faible plus l'arc sera mis à jour avec une grande quantité de phéromone. On peut maintenant définir le  $\Delta_{\tau_{ij}}(t)$  de la formule de mise à jour des phéromones ainsi :

$$\Delta_{\tau_{ij}}(t) = \sum_{k=1}^m \Delta_{\tau_{ij}^k}(t) \quad (2.15)$$

### Description de l'algorithme

Initialement :

1. les  $m$  fourmis sont réparties aléatoirement sur les  $n$  villes.
2. Pour chaque fourmi, la liste qui modélise sa mémoire contient sa ville de départ.
3. Les pistes de phéromones sont initialisées comme suit :  $t_{ij}(0) = c$ , où  $c$  est une petite constante positive, qui ne peut être nulle (sinon, il y a un problème lors du calcul de (2.12))

Après  $n$  itérations, nous sommes à l'instant  $t$ , toutes les fourmis ont terminé leur tour, chacune a une liste "mémoire" pleine et est revenue à sa propre ville de départ. À ce moment :

1. Chaque fourmi calcule sa valeur  $L_k(t)$ .
2. Variables  $\tau_{ij}^k(t)$  sont calculées conformément à la formule (2.12).
3. Les variables de phéromone  $\tau_{ij}(t)$  sont mises à jour suivant la formule (2.13). En d'autres termes, la fourmi refait son tour en sens inverse tout en déposant des phéromones.
4. On observe quelle fourmi a trouvé le tour de longueur minimum (i.e. on recherche la fourmi  $k$  telle que  $k = \min_{k=1}^m L_k(t)$ ). Si ce tour est meilleur que le meilleur tour jusqu'ici, on le mémorise
5. Les mémoires des fourmis (liste des villes visitées) sont effacées.
6. Les fourmis recommencent un nouveau tour, toujours au départ de la ville sur laquelle elles avaient été placées au début de l'algorithme.



**Fin de l'algorithme :** e. On arrête l'algorithme après un nombre de cycles égal à une constante  $NC_{max}$ . Si à partir d'un instant, toutes les fourmis font le même tour, l'algorithme s'interrompt : on est dans une situation de stagnation où le programme arrête de chercher des alternatives. L'algorithme donne en retour le meilleur tour mémorisé.

Comme on pourrai l'observer, l'algorithme des colonies de fourmis semble être solution à de nombreux problèmes complexes, souvent coûteux en terme de temps. Inspirée du comportement naturelle des fourmis, et adaptée à la base pour le problème du voyageur de commerce, il peut cependant devenir intéressant d'utiliser une telle approche pour un quelconque autre problème complexe.

Deuxième partie

Contribution

# Chapitre 3

## Proposition d'une approche de recommandation

### 3.1 Introduction

Ayant pour objectif d'augmenter les performances du filtrage collaboratif, nous proposons donc une amélioration de celui-ci. Notre approche sera basée sur un clustering optimisé dont l'objectif est de donner le meilleur partitionnement possible des utilisateurs en vue d'effectuer des recommandations plus précises.

### 3.2 Description de notre approche de recommandation

La figure 3.1 décrit notre approche de recommandation qui est basée sur un filtrage collaboratif de type user-user, que nous combinons avec une technique de clustering optimisé. Les étapes de notre approche peuvent être résumées comme suit :

- Utilisation des évaluations données par les utilisateurs sur les items (matrice d'usage)
- Partitionnement des utilisateurs en utilisant un clustering qui sera basé sur l'utilisation de l'algorithme des k-medoids. Ceci permettra de regrouper les utilisateurs en différents ensembles tels que les individus appartenant

à un même groupe soient considérés comme étant des utilisateurs voisins. Contrairement à certains algorithmes de clustering tels que k-means, l'algorithme des k-medoids est décrit comme étant moins sensible aux "*outliers*", ce qui nous aidera à identifier des groupements d'utilisateurs plus efficacement sans que les individus "bruyants" (ex : faux profils, évaluations infondées ou hasardeuses) n'affecte les résultats du clustering.

- Optimisation du partitionnement qui sera basée sur l'utilisation de la méta-heuristique des colonies de fourmis. Étant utilisée dans de nombreux problèmes d'optimisation, notamment pour celui du voyageur de commerce, cette méta-heuristique offre souvent de bons résultats.
- Prédiction qui sera basée sur le partitionnement issu des étapes précédentes.

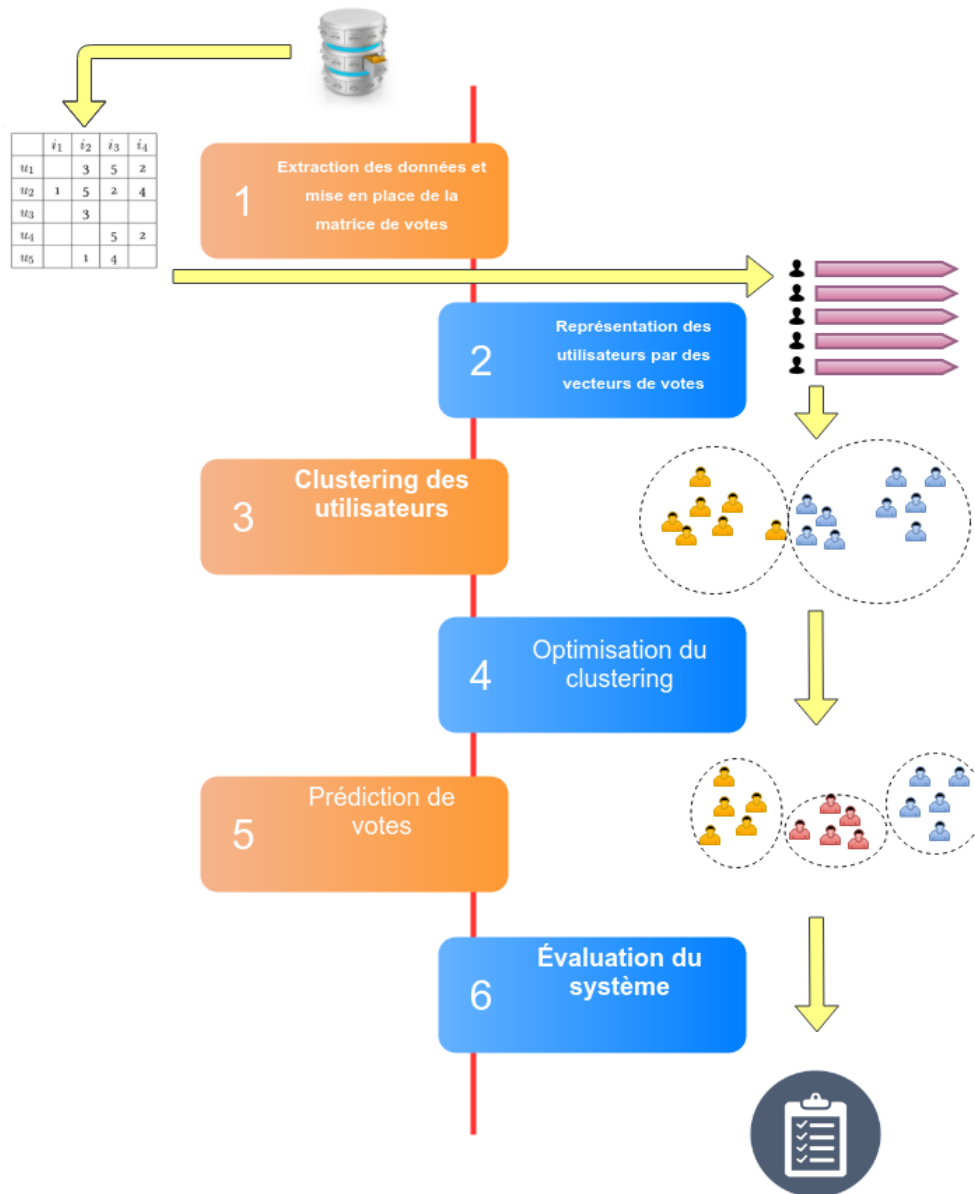


FIGURE 3.1 – Notre approche de recommandation d'items

### 3.2.1 Voisinage et clustering

#### Formalisation de l'approche de clustering

Afin qu'un système de recommandation basé sur le filtrage collaboratif puisse prédire l'évaluation d'un utilisateur sur un item donné, il est indispensable que celui-ci connaisse le voisinage de ce même utilisateur. C'est dans ce cadre, qu'une approche par clustering intervient afin d'identifier des groupes d'utilisateurs similaires. L'algorithme des K-Medoids sera utilisé pour cette tâche de clustering.

L'ensemble de données est donc défini par les utilisateurs, tels que ceux-ci soient représentés par des vecteurs de valeurs de même dimension ou chaque valeur est relative à l'appréciation de l'utilisateur courant sur l'item lié à l'indice de la case du vecteur.

Une évaluation peut se présenter sous différentes natures : valeurs entières, valeurs binaires, notes etc ... Étant donné que notre travail utilisera MovieLens comme base de test, le processus de clustering devra ainsi tenir compte de la structure du jeu de données (MovieLens). MovieLens est une plate-forme de recommandation de films, où chaque utilisateur note le film avec une valeur allant de 1 à 5, où 1 serait une note médiocre et 5 une très bonne note.

Ainsi, un vecteur décrivant un utilisateur représentera d'une certaine manière une ligne de la matrice d'usage, telle que la ligne fera référence à l'utilisateur et la colonne à l'item. La taille du vecteur est donc égale au nombre d'items. Les valeurs contenues dans les vecteurs seront comprises entre 1 et 5.

Par ailleurs, le processus de clustering aura pour fonctionnalité de regrouper les utilisateurs ayant des vecteurs plus ou moins similaires, c'est à dire ceux ayant plus ou moins noté les mêmes films de manière similaire, et seront donc considérés comme voisins.

#### Algorithme et illustration

K-Means est probablement l'un des algorithmes les plus utilisés en clustering, un fait qui est dû à son efficacité et notamment pour sa convergence rapide liée à une faible complexité. Cependant, K-Means est décrit comme étant très sensible aux *outliers*, et c'est dans cette perspective que k-medoids apporte une solution. Décrit comme étant une amélioration de K-Means, K-medoids reste cependant très similaire à son prédécesseur mais moins sensible aux données aberrantes.

Il est important de souligner que notre algorithme (PAM) sera enrichi par rapport à l'algorithme générique, afin d'être en mesure de mieux traiter les valeurs manquantes et les données aberrantes. La fonction de distance utilisée est la

distance de Manhattan.

Afin de pouvoir gérer le problème des valeurs manquantes, on a du apporter un léger changement à la formule de distance initiale, telle que si l'un des deux utilisateurs n'a pas noté l'item  $It_i$  ou que les deux ne l'ont pas fait, alors la différence sera maximisée à 4. Ce qui équivaut à dire que si deux utilisateurs ont très peu de films notés en commun, la distance les séparant sera relativement grande.

### Distance de Manhattan

$$d(A, B) = \sum_{i=1}^k |D_{i_A} - D_{i_B}| \quad (3.1)$$

tel que :

- $k$  = la dimension dans laquelle sont définis les points
- $D_i$  = la valeur du point à la dimension  $i$

### Distance de Manhattan modifiée

$$d(A, B) = \sum_{i=1}^k \begin{cases} |D_{i_A} - D_{i_B}| & \text{si } D_{i_A} \neq nul \text{ \& } D_{i_B} \neq nul \\ 4 & \text{sinon} \end{cases} \quad (3.2)$$

**Algorithmme : K-Medoids : pour le clustering de nos utilisateurs Entrées :**

1.  $k$  : le nombre de clusters.
2. L'ensemble des données
3.  $Th(threshold)$  : nombre minimale d'individus dans un cluster

**Sorties :**

1. medoids
2. Clusters de sortie

**méthode :**

- Choisir  $k$  points à partir de l'ensemble des données comme medoids initiaux
- **Répéter**
  - Pour chaque point, calculer la distance séparant celui-ci des différents medoids et ainsi assigner le point au cluster approprié en tenant compte de la distance la plus courte.
  - Sélectionner aléatoirement un point non-représentatif  $o_{random}$
  - Calculer le coût total,  $S$ , de l'échange du point représentatif (medoid) $o_j$  avec  $o_{random}$
  - Si  $S < 0$  alors permuter  $o_j$  avec  $o_{random}$  pour former le nouvel ensemble des  $k$  points représentatifs (medoids)

**Jusqu'à :** Aucun changement **Si :** Il existe des clusters avec un nombre de points inférieur à  $TH$  **Alors :** Assigner tout ces points (medoids compris) au cluster qui leur est le plus proche

**FIN**

**Remarque :**

Ainsi, le traitement supplémentaire effectué à la fin du processus de clustering peut conduire à nombre de clusters générés inférieurs à celui indiqué comme paramètre d'entrée, comme il est illustré dans le schéma ci-dessous :



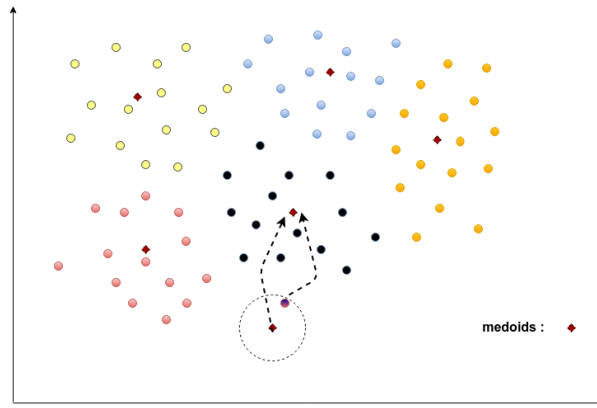


FIGURE 3.2 – Clustering

### 3.2.2 Optimisation du clustering

#### Adaptation de la méta-heuristique

Les algorithmes tels que k-means ou k-medoids sont des algorithmes à convergence rapide avec un résultat fortement lié aux centroïdes/medoids choisis initialement, ce qui est basé sur l'aléatoire.

Dans cette section, nous allons tenter d'optimiser la qualité du clustering, en nous basant sur l'algorithme des colonies de fourmis.

Dans notre cas, optimiser le clustering reviendrait donc à trouver les medoids de départ qui permettraient d'atteindre le meilleur clustering possible. Ainsi, en analogie au comportement des fourmis, nos agents devront partir à la recherche de ressources qui seraient en fait des combinaisons de points faisant partie de notre ensemble de données, et que considérera l'algorithme de clustering k-medoids comme ensemble de medoids de départ.

Au départ, il est clair que nos fourmis devront aller vers la recherche de ressource de façon aléatoire. Dans ce cas de figure, les combinaisons de points de départ trouvées par chaque fourmi seront composées de points pris de manière hasardeuse, comme l'illustre le schéma suivant :

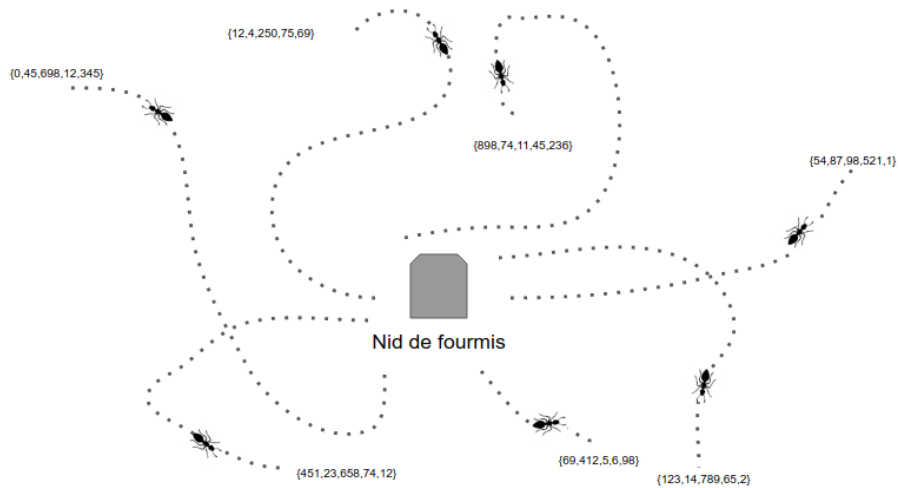


FIGURE 3.3 – Éparpillement initial des fourmis de manière aléatoire

Plus un ensemble de points permet la convergence vers un clustering de qualité, plus la fourmi ayant trouvé cette solution déposera de phéromone sur l'ensemble de ces points. Ainsi, la table de phéromone, initialement complétée de valeurs nulles pour chaque point de notre ensemble de medoids, sera mise à jour telle que chaque point des solutions trouvées sera associé à une certaine quantité de phéromone proportionnelle à la qualité de l'ensemble de départ auquel elle appartient.

0	minPH		0	minPh + 0.089
1	minPh		1	minPh + 0.78
2	minPh		2	minPh + 0.854
3	minPh		3	minPh
4	minPh		4	minPh
5	minPh		5	minPh + 0.911
6	minPh		6	minPh + 0.91
7	minPh		7	minPh
8	minPh		8	minPh
...	...		...	...

FIGURE 3.4 – Première mise à jour de la table de phéromones

Comme le montre la transition illustrée sur la figure 3.4 : Étant donné que les points 0, 1, 2, 5 et 6 font partie dans notre exemple des solutions trouvées initialement par nos agents, les quantités de phéromone qui leur sont associés seront incrémentées proportionnellement à la qualité des solution auxquelles ils appartiennent.

De ce fait lors de l'itération suivante, l'ensemble de la colonie traitera très probablement des combinaisons contenant des points riches en phéromone. Il s'agit là du processus de transition, ou chaque fourmi, devra constituer une solution en partant à la recherche de points qui lui permettront de la constituer.

Lors de cette étape, chaque fourmi constituera une solution en prenant les points qui sont les plus susceptibles de constituer une solution de bonne qualité. Pour ce faire, un paramètre important " $q_0$ " jouera un rôle essentielle dans l'aspect aléatoire du processus.

Le processus de construction de solution se fera par une intégration point par point dans l'ensemble. Au moment où la fourmi cherchera à intégrer un point dans sa solution, c'est à dire choisir la suite de l'itinéraire à emprunter, une valeur sera tiré aléatoirement telle que celle-ci sera comprise entre 0 et 1. Si  $q_0$  est inférieur à cette valeur, alors il faudra insérer le meilleur point parmi ceux non déjà choisis, à partir de la table de phéromone. Dans le cas contraire, il faudra choisir un point de manière aléatoire entre ceux présents dans notre ensemble de points.

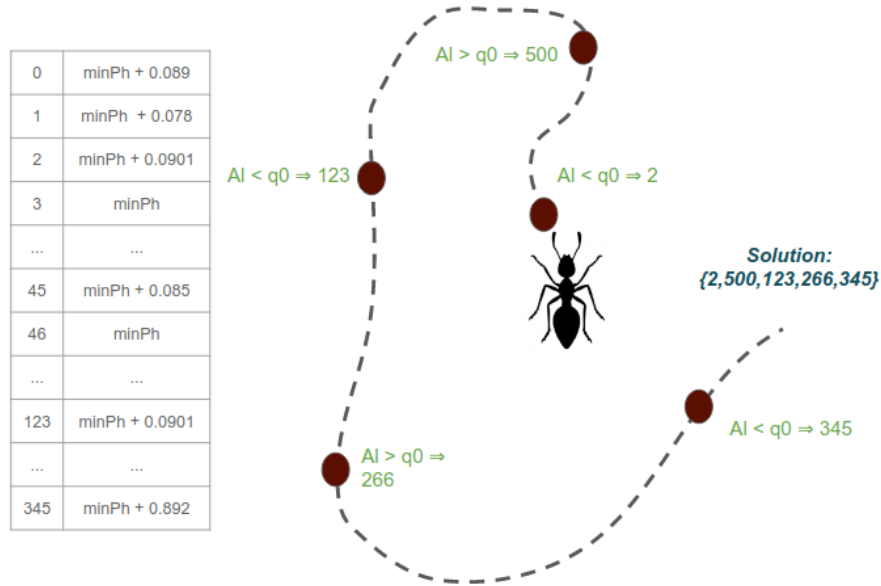


FIGURE 3.5 – Construction de nouvelles solutions

En effet, les solutions construites par chaque fourmi ne devront pas être constituées forcément des meilleurs points, c'est à dire des points ayant le plus de phéromones. Les solutions doivent être constituées de points de qualité et de points pris de façon aléatoire afin d'éviter que les fourmis ne se dirigent toutes vers la même ressource, et ainsi intégrer un aspect aléatoire au processus de recherche de nourriture, comme c'est le cas dans le comportement naturelle des fourmis.

Ainsi, après chaque cycle, comme c'est le cas en réalité, il y a une évaporation de la phéromone se trouvant dans l'espace de recherche. Le taux d'évaporation sera ainsi fixé avant le lancement de l'algorithme.

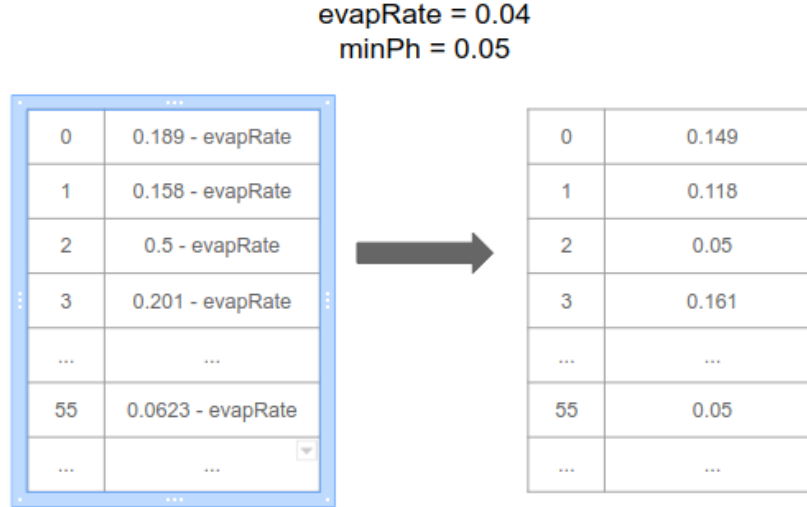


FIGURE 3.6 – Évaporation de phéromone après chaque cycle

Nous utiliserons le paramètre *MinMax* qui permettra à un point de ne pas contenir de phéromone au-delà ou en deçà de certains seuils. Ceci est illustré par la variable *minPh* par exemple sur la figure 3.6. Ainsi, la quantité de phéromone présente sur un point ne sera jamais inférieure à la valeur de *minPh*.

D'autre part, on devra limiter la durée d'exécution de l'algorithme si celui-ci ne parvient pas à obtenir la solution de qualité désirée. Pour cela, un paramètre fixant le nombre maximal d'itérations sera introduit.

La qualité d'une solution sera évaluée grâce à une fonction d'évaluation basée sur la distance, que nous avons proposée et qui est décrite comme suit :

$$Fit(Solution) = \sum_{i=1}^k \sum_{x \in C_i} dist(x, C_i) \quad (3.3)$$

Cette fonction étant basée sur la distance devra donc être minimisée, ainsi une petite distance équivaudrait à une forte sécrétion de phéromone.

## Algorithme et illustration

Voici l'énoncé de l'algorithme de clustering optimisé par colonie de fourmis :

**Algorithme :** **Clustering optimisé :** *Optimisation par colonie de fourmis*

**Entrées :**

1. Nb : Nombre de fourmis.
2. MaxIter : Nombre maximal d'itérations (critère d'arrêt)
3.  $q_0$  : paramètre de transition
4. MaxPh, MinPh : Quantités maximales et minimales de phéromone pouvant être associées à un point
5. evapRate : taux d'évaporation

**Sorties :**

1. medoids
2. Clusters de sortie

**méthode :**

- Initialiser la table de phéromone, en attribuant à chaque point de l'ensemble de données une quantité de phéromone égale à MinPh
- **Répéter**
  - **Pour :** Chaque fourmi
    - Al = nombre aléatoire  $\in [0, 1]$
    - **Si :**  $q_0 < Al$   
**Alors :** Inclure dans la solution le point ayant le plus de phéromone et non déjà choisi **Sinon :** Ajouter n'importe quel point dans la solution
    - Appeler k-medoids avec la solution générée
    - Évaluer la solution générée
    - Sauvegarder les résultats
  - Considérer la meilleure solution issue de l'exploration précédente
  - **Si :** Il y a une amélioration globale **Alors :** Ajouter des phéromones aux points de cette solution sans dépasser maxPh
  - **Pour :** Chaque point de l'ensemble de données **Faire :** Faire évaporer la phéromone selon *evapRate* sans descendre en deçà de *MinPh*
- Jusqu'à :** Atteindre MaxIter (*Retenir la meilleure solution*)

**FIN**

Il est important d'indiquer que certains aspects de l'algorithme proposé ont été réfléchis et appliqués à l'algorithme générique dans le but de pouvoir adapter celui-ci au problème qui nous est posé. Voici les principales spécificités liées à

notre algorithme :

### **Un seul critère d'arrêt**

Généralement, on retrouve deux critères d'arrêt pour l'algorithme des colonies de fourmis : le nombre d'itérations maximale et la qualité désiré. Cependant, le second critère ne sera pris en compte car nous ne désirons pas atteindre une qualité connue d'avance mais plutôt à observer les performances de l'approche et donc la meilleure qualité que celle-ci peut apporter durant un laps de temps défini.

### **Modification du concept de transition**

Ce concept est présent dans l'algorithme proposé par Dorigo et dans de nombreuses versions issues de l'originale. Cependant, le concept de transition reste notamment adapté au problème du voyageur de commerce, et son utilisation dans notre contexte ne peut s'effectuer de la même manière. Ainsi, pour garder le coté aléatoire de l'approche nous utilisons un paramètre  $q_0$  qui fera en sorte que les fourmis ne prennent pas toujours la meilleure solution, ce qui éviterait une stagnation et une convergence trop rapide.

### **Choix de la solution**

Il s'agit probablement de l'aspect le plus important de l'algorithme. En observant le fonctionnement de l'algorithme des k-medoids, on peut facilement voir que celui-ci converge assez rapidement et ne s'éloigne que très peu de la solution aléatoire de départ. Par conséquent, nous avons pensé à faire de cette ensemble aléatoire de points de départ notre solution. En jouant sur celle-ci grâce à notre méta-heuristique, on parviendra à trouver une combinaison de départ permettant de converger vers une solution de qualité.

### **La prédiction**

C'est une fois que le clustering est effectué qu'il sera possible d'émettre les prédictions. Celle-ci s'appuie sur une fonction mathématique basée sur la notion de voisinage. Il est primordial de mettre en place la matrice des votes et d'identifier les groupes d'utilisateurs voisins entre eux et calculer la similarité qui les relie.

### La fonction de prédiction

Voici la fonction de prédiction qu'utilisera notre système [28] :

$$Pred(u_i, i_k) = \bar{r}(u_i) + \frac{\sum_{u_j \in V_i} \gamma(u_i, u_j) \times (r_{u_j, i_k} - \bar{r}(u_j))}{\sum_{u_j \in V_i} \gamma(u_i, u_j)} \quad (3.4)$$

tel que :

- $\gamma(u_i, u_j)$  : est la similarité entre un utilisateur  $u_i$  et son voisin  $u_j$ , tel que  $u_i \in V_i$

### Le fonction de corrélation

Pour la fonction de corrélation, nous avons opté pour la corrélation de Pearson, qui est défini comme suit :

$$pearson(u, v) = \frac{\sum_{i \in I} (r_{u,i} - \bar{r}_u) \cdot (r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I} (r_{u,i} - \bar{r}_u)^2 \cdot \sum_{i \in I} (r_{v,i} - \bar{r}_v)^2}} \quad (3.5)$$

tel que :

- $r_{u,i}$  est l'estimation de l'utilisateur  $u$  sur l'item  $i$ .
- $r_{v,i}$  est l'estimation de l'utilisateur  $v$  sur l'item  $i$ .
- $\bar{r}_u$  est la moyenne de toutes les notes de l'utilisateur  $u$ .
- $\bar{r}_v$  est la moyenne de toutes les notes de l'utilisateur  $v$ .
- $I$  est l'ensemble des items

## 3.3 Conclusion

Comme nous avons pu le voir, la conception de notre système inclut un certain nombre de procédures aussi indispensables les unes que les autres.

Ayant déjà été utilisé dans de nombreux travaux autour des systèmes de recommandation, le clustering d'utilisateurs peut permettre d'améliorer les prédictions. Cependant, apporter une optimisation à ce processus, comme nous l'avons fait via une méta-heuristique, devrait grandement relever les performances de l'approche que nous avons proposée.



Le prochain chapitre sera consacré à l'implémentation et à l'évaluation des différents algorithmes présentés dans notre approche de recommandation (Algorithme de FC basé sur un clustering, algorithme de FC basé sur un clustering optimisé) et qui seront comparés avec des algorithmes classique de filtrage collaboratif ainsi qu'avec d'autres travaux liés.

# Chapitre 4

## Implémentation et évaluation

### 4.1 Introduction

Dans le chapitre précédent, nous avons modélisé notre approche et effectué une description claire de la technique proposé dans notre travail.

Dans cette partie, nous parlerons plus en détails sur ce qui a été développé. Nous commencerons par présenter notre environnement de travail en citant langage, outils et bibliothèques utilisés. Par la suite nous présenterons l'application réalisée, en décrivant son fonctionnement et en explicitant son mode d'emploi. Pour finir, nous analyserons les performances de l'approche développée en effectuant des tests et en comparant ces derniers avec ceux d'autres techniques également implémenté et exploitables directement via notre application.

## 4.2 Architecture technique du système

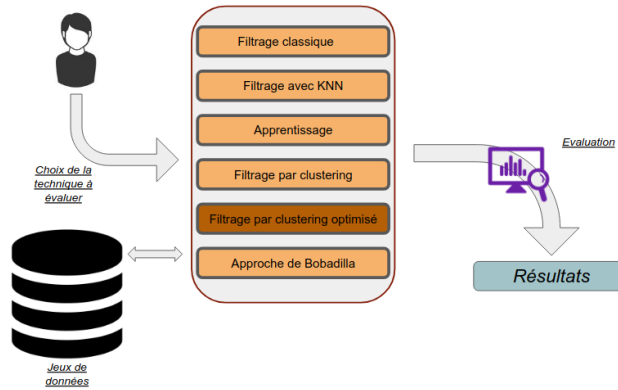


FIGURE 4.1 – Schématisation de l'architecture du système

## 4.3 Environnement de travail

Nous allons présenter dans cette partie, tout les outils qui nous aurons permis de réaliser notre application.

### 4.3.1 Langage de programmation *Python*

#### Présentation du langage

Créé originellement par le programmeur Guido van Rossum autour de 1990, contrairement à de nombreux langages qui sont supportés par des sociétés, la nature open source du langage Python lui permet d'évoluer grâce à la communauté des développeurs Python. Une mise à jour importante a été le passage de la version 2 à la version 3 [54].

#### Caractéristiques du langage et avantages

- Python se distingue notamment par une syntaxe simplifiée et concise, celle-ci permet notamment un gain de temps considérable au programmeur.
- On y trouve des types de données évolués (listes, dictionnaires,...), ce qui permet à Python d'être à la fois très compacts et très lisibles.



FIGURE 4.2 – Logos des packages *Python* utilisés

- Il est parfois considéré comme un langage réservé au script, à tort, car il offre un grand confort dans le développement d'applications de divers types [54].
- Python est orienté-objet. Il supporte l'héritage multiple et la surcharge des opérateurs
- Python est un langage multi-plateforme, il fonctionne aussi bien sous Windows, Mac, Linux ainsi que sous de nombreuses variantes dérivées d'UNIX
- Python est un langage open-source. la grande communauté de contributeurs qui est derrière son développement lui permet d'être en constante évolution
- Une mise à jour importante a été le passage de la version 2 à la version 3.

### 4.3.2 Les principaux packages *Python* utilisés

#### NumPy

NumPy est le package fondamental pour l'informatique scientifique avec Python. Il contient entre autres un puissant objet *array* multi-dimensionnel et offre plusieurs possibilités pour les traitements algébriques.

#### Scikit-learn

Ce package *Python* procure des outils simples et efficaces pour l'exploration de données et l'analyse de données. On y trouve un grand nombre d'algorithmes prédéfinis pour la classification et pour le clustering.

## **Pandas**

Pandas est un package *Python* fournissant des structures de données rapides, flexibles et expressives conçues pour rendre le travail avec des données «relationnelles» ou «étiquetées» à la fois facile et intuitif. Il vise à être le bloc de construction fondamental de haut niveau pour effectuer des analyses de données réelles et pratiques en Python [4].

## **Maths**

Ce package permet d'effectuer de nombreuses opérations mathématiques en un minimum d'instructions.

## **time**

Ce package offre la possibilité de manipuler l'aspect date et temps dans nos scripts. Il permet entre autres d'estimer les temps d'exécutions des algorithmes implémentés.

## **random**

random est un module qui permet de jouer avec le hasard et l'aléatoire, souvent utiles pour le développement de méta-heuristiques.

### **4.3.3 Outils et logiciels**

#### **Système d'exploitation (Ubuntu - GNU/Linux)**

Ubuntu est un système d'exploitation GNU/Linux basé sur la distribution Linux Debian. Il est développé, commercialisé et maintenu pour les ordinateurs individuels par la société Canonical [53].

Ubuntu se définit comme « un système d'exploitation utilisé par des millions de PC à travers le monde » et avec une interface « simple, intuitive, et sécurisée ». Elle est la distribution la plus consultée sur Internet d'après le site Alexa, et le système d'exploitation le plus utilisé sur les systèmes Cloud ainsi que sur les serveurs informatiques [53].



FIGURE 4.3 – Logos des outils utilisés

### **L'environnement de développement intégré (IDE) Pycharm**

PyCharm est un environnement de développement intégré (abrégé EDI en français ou en anglais : IDE (Integrated Development Environment)) utilisé pour programmer en Python [53].

Il offre l'analyse de code, un débogueur graphique, la gestion des tests unitaires, l'intégration de logiciel de gestion de versions, et supporte le développement web avec Django [53].

Il est développé par l'entreprise tchèque JetBrains. Il est multi-plateforme et fonctionne sous Windows, Mac OS X et Linux. Il est décliné en édition professionnelle, réalisé sous licence propriétaire, et en édition communautaire réalisé sous licence Apache [53].

### **Gestionnaire de versions *Git***

Git est un logiciel de gestion de versions décentralisé. C'est un logiciel libre créé par Linus Torvalds, auteur du noyau Linux, et distribué selon les termes de la licence publique générale GNU version 2. En 2016, il s'agit du logiciel de gestion de versions le plus populaire qui est utilisé par plus de douze millions de personnes [53].

### **Editeur de texte : Sublime Text**

Sublime Text est un éditeur de texte générique codé en C++ et Python, disponible sur Windows, Mac et Linux. Il intègre de nombreux plugins pour la programmation ainsi que pour la rédaction (en latex par exemple).

## 4.4 présentation de l'application

### 4.4.1 Vue générale

L'application que nous avons développée permettra de lancer les différents algorithmes implémentés et de visualiser en détails les performances et les résultats de chaque technique.

il sera également possible de choisir le data-set (jeux de données) à utiliser. Dans un premier temps, certaines techniques implémentées ne seront pas généralisées pour traiter n'importe quelle structure de data-set. Cependant, la modularité du code source de l'application permettra facilement d'apporter des mises à jours aux algorithmes afin que ceux-ci puissent être testés avec d'autres data-set.

En tout premier lieu, on doit donc choisir notre jeux de données via une première fenêtre qui s'affiche juste après le lancement de l'application :

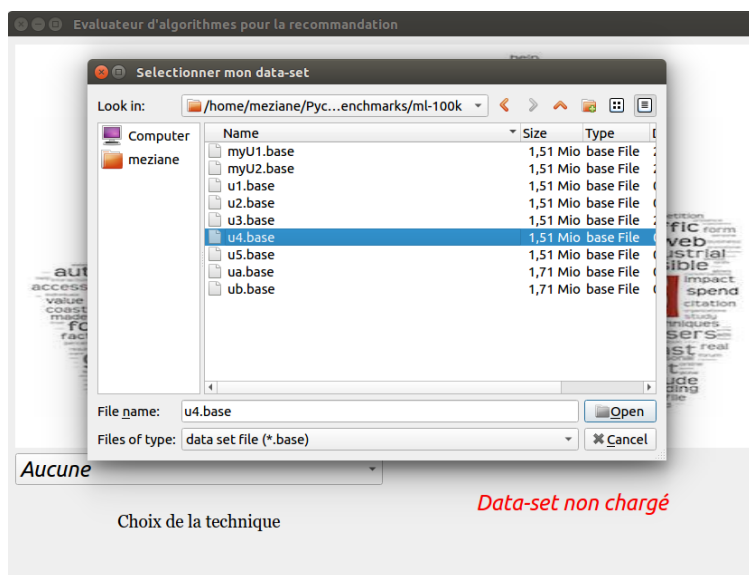


FIGURE 4.4 – Choix du data-set

Par la suite, il sera possible de tester l'un des algorithmes disponibles sur la l'application.

- **Aucune** : Il est possible de ne sélectionner aucune technique spécifique, cette approche est basé sur la notion de voisinage qui sera établie selon une fonction de corrélation et en prenant comme voisins les individus ayant



FIGURE 4.5 – Choix de l’algorithme à tester - Fenêtre d’accueil

une corrélation avec l’individu en question supérieure à un certain seuil qu’il sera possible de fixer.

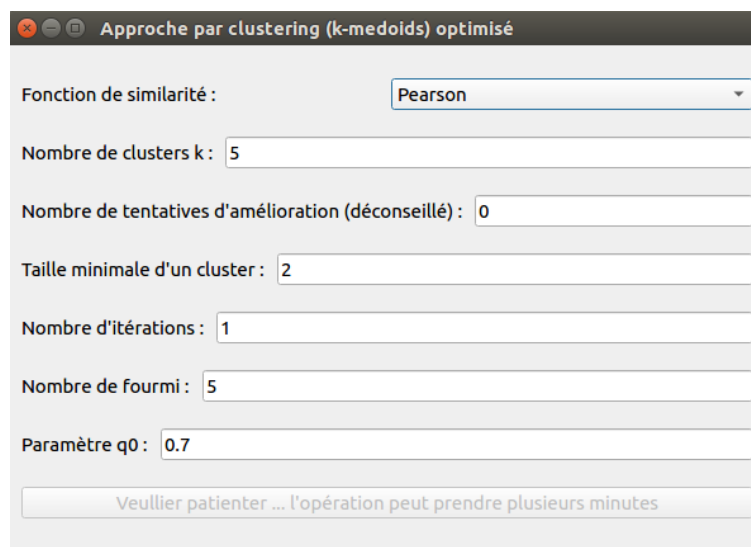
- **KNN** : Il s’agit d’appliquer la technique des k plus proches voisins afin de déterminer le voisinage de chaque individu. La partie "prédiction" sera effectué de la même façon que celle appliqué pour notre approche.
- **Apprentissage** : Il est possible via notre application, d’effectuer des prédiction sur approche basé modèle. Cela pourra se faire suite à un apprentissage effectué selon la technique d’apprentissage supervisé choisie.
- **Clustering** : Cette approche fut déjà présentée et proposée dans la littérature. Elle est basé sur un filtrage collaboratif ou le voisinage est déterminé suite à la mise en place de groupes d’individus issus d’un clustering, les techniques de clustering proposés sont k-means et k-medoids. L’algorithme k-medoids sera cependant personnalisé comme il nous l’avons mentionnée dans le chapitre précédant. il sera ainsi plus adapté au problème de la recommandation basée sur le filtrage collaboratif.
- **Clustering optimisé** : Il s’agit de l’approche que nous avons développée et que nous allons comparer avec l’ensemble des autres approches implémentées.
- **Technique de Bobadilla** : Proposé par Bobadilla and all dans l’article [30], cette approche sera notre principale baseLine, avec laquelle nous comparerons nos résultats. Nous avons ainsi pu implémenter cette technique afin de l’évaluer par rapport au data-set de notre choix.



#### 4.4.2 Description de l'application par visualisation d'un cas d'utilisation

Comme décrit précédemment, l'utilisateur devra spécifier la technique à évaluer et le jeu de données à utiliser juste après le lancement de l'application.

Après cela, dépendamment de la technique choisie, l'utilisateur devra fixer les nombreux paramètres associés à l'algorithme à évaluer. Il sera par la suite possible de Lancer l'algorithme.



The screenshot shows a software window titled "Approche par clustering (k-medoids) optimisé". It contains several input fields for configuring a clustering algorithm. The "Fonction de similarité" is set to "Pearson" via a dropdown menu. The "Nombre de clusters k" is set to 5, "Nombre de tentatives d'amélioration (déconseillé)" is 0, "Taille minimale d'un cluster" is 2, "Nombre d'itérations" is 1, "Nombre de fourmi" is 5, and "Paramètre q0" is 0.7. At the bottom, a status bar displays the message "Veullier patienter ... l'opération peut prendre plusieurs minutes".

Paramètre	Valeur
Fonction de similarité	Pearson
Nombre de clusters k	5
Nombre de tentatives d'amélioration (déconseillé)	0
Taille minimale d'un cluster	2
Nombre d'itérations	1
Nombre de fourmi	5
Paramètre q0	0.7

FIGURE 4.6 – Choix des paramètres

À la fin du traitement, une fenêtre affichant tous les détails du processus fera irruption et fournira toutes les données, informations, et résultats liés au test réalisé. l'utilisateur sera également en mesure d'exporter ces résultats dans un fichier ".csv" afin de pouvoir revoir ou réutiliser ultérieurement ces informations.

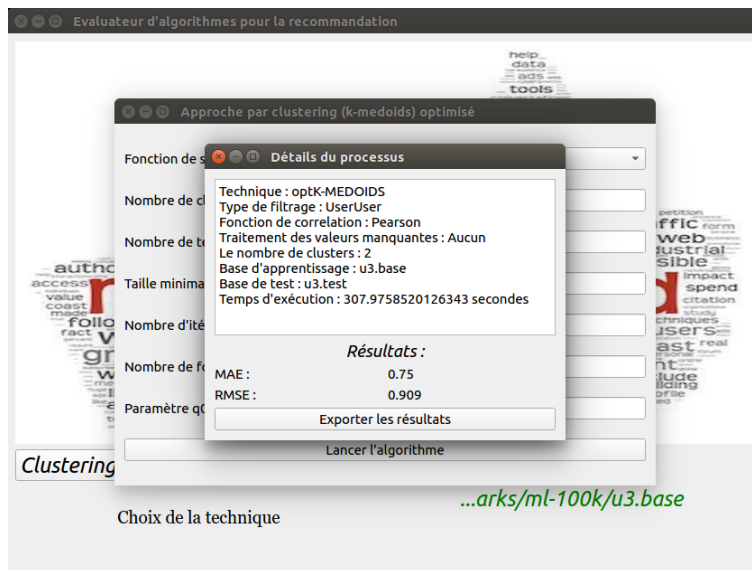


FIGURE 4.7 – visualisation des résultats

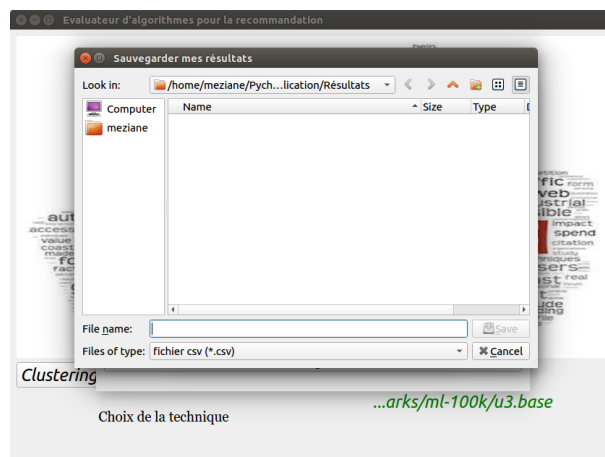


FIGURE 4.8 – Sauvegarde des résultats

## 4.5 Expérimentations

### 4.5.1 Jeux de données (base de test)

#### Présentation de MovieLens

MovieLens est un site qui vous aide à trouver les films que vous êtes susceptible d'apprécier. En évaluant les différents films présent sur sa plate-forme, MovieLens crée un profil de goût personnalisé, puis vous recommande d'autres films à regarder. MovieLens est géré par GroupLens, un laboratoire de recherche de l'Université du Minnesota. GroupLens Research a collecté et mis à disposition des ensembles de données d'évaluation à partir du site Web MovieLens. Les ensembles de données ont été collectés sur différentes périodes, en fonction de la taille de l'ensemble.

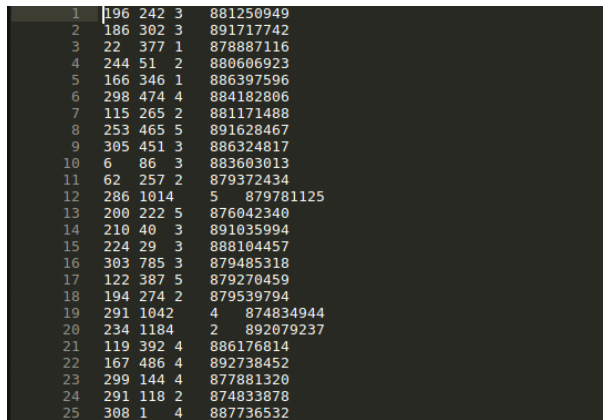
Des data-sets de différentes tailles sont ainsi proposé. On retrouve des jeux de données allant de 100 000 évaluations jusqu'à 20 millions d'évaluations, même si ces derniers sont plus destinées à la recherche ou de grand moyens matériels sont souvent nécessaires.

Étant limité par le temps et par la puissance de calcul de nos machines, nous opterons donc dans un premier temps pour des data-set de 100 000 évaluations afin d'effectuer nos tests

#### Architecture du data-set

Un dossier (data-set MovieLens) se présente sous la structure suivante :

- Un fichier "u.data" de 100 000 lignes, contenant l'ensemble des évaluations (l'id de l'utilisateur, l'id de l'item, l'évaluation (de 1 à 5), la date d'évaluation)



1	196	242	3	881250949
2	186	302	3	891717742
3	22	377	1	878887116
4	244	51	2	880606923
5	166	346	1	886397596
6	298	474	4	884182806
7	115	265	2	881171488
8	253	465	5	891628467
9	305	451	3	886324817
10	6	86	3	883603013
11	62	257	2	879372434
12	286	1014	5	879781125
13	200	222	5	876042340
14	210	40	3	891035994
15	224	29	3	888104457
16	303	785	3	879485318
17	122	387	5	879270459
18	194	274	2	879539794
19	291	1042	4	874834944
20	234	1184	2	892079237
21	119	392	4	886176814
22	167	486	4	892738452
23	299	144	4	877881320
24	291	118	2	874833878
25	308	1	4	887736532

FIGURE 4.9 – structuration du fichier des évaluations du data-set

- Un fichier "u.genre" ou on peut retrouver les différents genres des films présents sur la plate-forme.
- Un fichier "u.item" où sont indiqués tous les films avec l'ensemble des informations qui leur sont relatives : date de sortie, genre etc ...
- Un fichier "u.occupation" indiquant les différentes professions des utilisateurs
- Un fichier "u.user" contenant les informations relatives à chaque utilisateur : âge, sexe, profession etc ...

## Exploitation du data-set

Notre approche se base sur le filtrage collaboratif mais ne se base en aucun sur le contenu sémantique. Ainsi, de nombreuses informations contenues dans le data-set seront ignorées. Nous n'aurons donc besoin que des identifiants des utilisateurs et des items ainsi que des différentes évaluations.

Le processus de test et d'évaluation d'un algorithme nécessite deux ensembles de données : un ensemble d'apprentissage et un ensemble de test.

En effet, l'ensemble de données contient en tout 100 000 évaluations, nous prendrons donc 80 000 pour l'apprentissage et 20 000 pour la prédiction et le calcul des métriques.

### 4.5.2 Choix des métriques utilisées

Nous nous baserons sur deux métriques afin d'évaluer notre travail : MAE et RMSE. MAE et RMSE sont des métriques permettant de calculer la différence

entre deux variables continues. En effet, ces deux métriques restent les plus utilisées dans le domaine et les plus pertinentes quant à évaluer la qualité d'un système de recommandation.

**MAE (mean absolute error)**

$$MAE = \frac{\sum_{i=1}^n |x_i - y_i|}{n} \quad (4.1)$$

**RMSE (root mean square error)**

$$RMSE = \frac{\sum_{i=1}^n |x_i - y_i|^2}{n} \quad (4.2)$$

où  $x$  et  $y$  sont les deux points concernés et  $n$  la dimension dans laquelle sont définis ces mêmes points.

### 4.5.3 Évaluations

Dans cette partie, nous réaliserons un ensemble de tests portant sur les performances de notre approche. Cependant, nous testerons également d'autres approches que nous avons implémentées afin d'effectuer un travail comparatif entre celles-ci et l'approche que nous avons proposée.

#### Filtrage classique

Dans cette phase d'expérimentations, nous allons évaluer la qualité de la recommandation lorsque celle-ci est basée sur un filtrage classique.

Nous ferons des expérimentations par rapport à la corrélation de Pearson, étant donné que celle-ci reste l'une des plus connues et des plus utilisées.

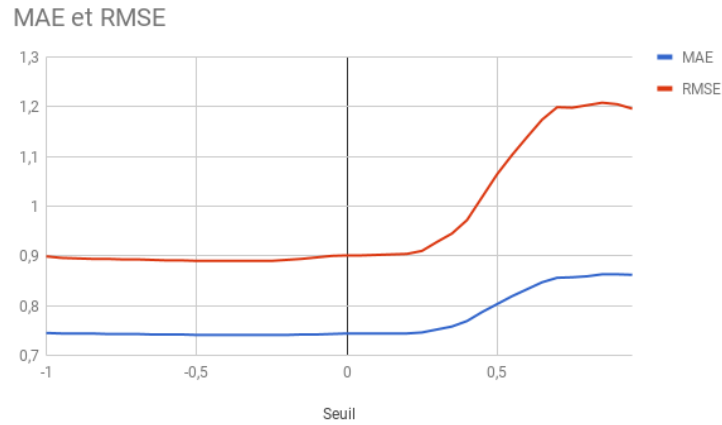


FIGURE 4.10 – variation des résultats selon un filtrage classique en fonction du seuil choisi

Seuil	-1.0	-0.95	-0.9	-0.85	-0.8	-0.75	-0.7	-0.65	-0.6	-0.55
MAE	0.745	0.744	0.744	0.744	0.743	0.743	0.743	0.742	0.742	0.742
Seuil	-0.5	-0.45	-0.4	-0.35	-0.3	-0.25	-0.2	-0.15	-0.1	-0.05
MAE	0.741	0.741	0.741	0.741	0.741	0.741	0.741	0.742	0.742	0.743
Seuil	0	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.45
MAE	0.744	0.744	0.744	0.744	0.744	0.746	0.752	0.758	0.769	0.787
Seuil	0.5	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9	0.95
MAE	0.803	0.819	0.833	0.847	0.856	0.857	0.859	0.863	0.863	0.862

TABLE 4.1 – Données associées au graphe 4.10 (MAE)

Seuil	-1.0	-0.95	-0.9	-0.85	-0.8	-0.75	-0.7	-0.65	-0.6	-0.55
RMSE	0.899	0.896	0.895	0.894	0.894	0.893	0.893	0.892	0.891	0.891
Seuil	-0.5	-0.45	-0.4	-0.35	-0.3	-0.25	-0.2	-0.15	-0.1	-0.05
RMSE	0.890	0.890	0.890	0.890	0.890	0.890	0.892	0.894	0.897	0.900
Seuil	0	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.45
RMSE	0.901	0.901	0.902	0.903	0.904	0.910	0.928	0.945	0.972	1.018
Seuil	0.5	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9	0.95
RMSE	1.064	1.103	1.139	1.174	1.199	1.198	1.203	1.208	1.205	1.196

TABLE 4.2 – Données associées au graphe 4.10 (RMSE)

En analysant le graphe ci-dessus, il est clair qu'au delà d'un certain seuil, environ 0.25, la qualité de la prédiction se détériore. Cependant il y a une

stagnation lorsque le seuil devient trop petit.

La meilleure valeur du seuil à choisir dans ce cas est de 0.25 afin de ne pas se retrouver avec des individus ayant un trop grand nombre de voisins inutilement étant donné qu'un petit nombre pourrait suffire et permettrait d'obtenir des résultats similaires.

Pour MAE, la meilleure valeur obtenue est de : 0.741

### Filtrage classique basé sur KNN

Pour cette approche, nous ferons varier la valeur de  $k$  et essayer de déduire laquelle de ses valeurs nous permettra d'atteindre les meilleurs résultats.

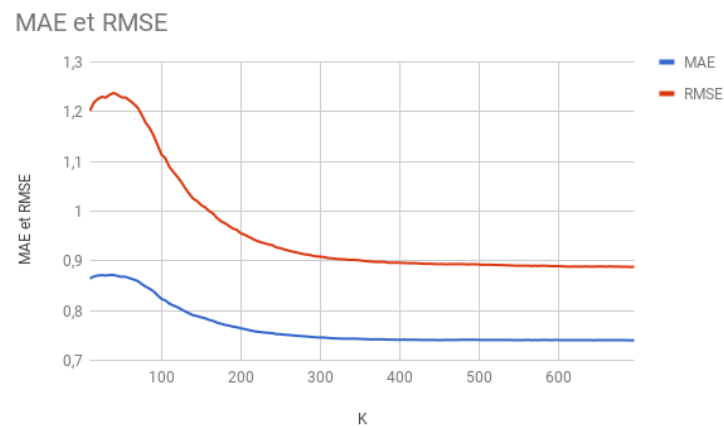


FIGURE 4.11 – variation des résultats selon un filtrage classique basé sur KNN en fonction de paramètre  $k$

K	10	15	20	25	30	35	40	45	50	55
MAE	0.864	0.868	0.870	0.87	0.870	0.871	0.871	0.869	0.867	0.867
K	60	65	70	75	80	85	90	95	100	105
MAE	0.865	0.862	0.859	0.853	0.848	0.844	0.838	0.831	0.823	0.820
K	110	115	120	125	130	135	140	145	150	155
MAE	0.814	0.810	0.807	0.802	0.798	0.794	0.790	0.789	0.786	0.784
K	160	165	170	175	180	185	190	195	200	205
MAE	0.781	0.779	0.775	0.773	0.771	0.769	0.767	0.766	0.764	0.762
K	210	215	220	225	230	235	240	245	250	255
MAE	0.760	0.759	0.757	0.756	0.755	0.755	0.754	0.752	0.752	0.751
K	260	265	270	275	280	285	290	295	300	305
MAE	0.751	0.750	0.749	0.748	0.748	0.747	0.746	0.746	0.746	0.745
K	310	315	320	325	330	335	340	345	350	355
MAE	0.745	0.744	0.743	0.743	0.743	0.743	0.743	0.743	0.743	0.742
K	360	365	370	375	380	385	390	395	400	405
MAE	0.742	0.742	0.742	0.742	0.742	0.741	0.741	0.741	0.741	0.741
K	410	415	420	425	430	435	440	445	450	455
MAE	0.741	0.741	0.741	0.741	0.741	0.740	0.740	0.740	0.740	0.740
K	460	465	470	475	480	485	490	495	500	505
MAE	0.740	0.741	0.741	0.741	0.741	0.741	0.741	0.741	0.741	0.740
K	510	515	520	525	530	535	540	545	550	555
MAE	0.741	0.741	0.741	0.740	0.740	0.740	0.740	0.740	0.740	0.740
K	560	565	570	575	580	585	590	595	600	605
MAE	0.741	0.740	0.740	0.740	0.740	0.740	0.740	0.740	0.740	0.740
K	610	615	620	625	630	635	640	645	650	655
MAE	0.740	0.740	0.740	0.740	0.740	0.740	0.740	0.740	0.740	0.740
K	660	665	670	675	680	685	690	695		
MAE	0.740	0.740	0.740	0.740	0.740	0.740	0.740	0.740		

TABLE 4.3 – Données associées au graphe 4.11 (MAE)



K	10	15	20	25	30	35	40	45	50	55
RMSE	1.201	1.218	1.225	1.229	1.228	1.234	1.237	1.233	1.228	1.228
K	60	65	70	75	80	85	90	95	100	105
RMSE	1.222	1.216	1.208	1.194	1.178	1.167	1.152	1.133	1.114	1.106
K	110	115	120	125	130	135	140	145	150	155
RMSE	1.089	1.078	1.069	1.058	1.046	1.035	1.024	1.020	1.012	1.007
K	160	165	170	175	180	185	190	195	200	205
RMSE	1.000	0.995	0.986	0.979	0.975	0.970	0.964	0.962	0.955	0.952
K	210	215	220	225	230	235	240	245	250	255
RMSE	0.948	0.944	0.940	0.937	0.935	0.933	0.931	0.926	0.925	0.922
K	260	265	270	275	280	285	290	295	300	305
RMSE	0.920	0.918	0.916	0.915	0.913	0.912	0.910	0.909	0.908	0.907
K	310	315	320	325	330	335	340	345	350	355
RMSE	0.905	0.905	0.904	0.903	0.903	0.902	0.902	0.901	0.900	0.899
K	360	365	370	375	380	385	390	395	400	405
RMSE	0.899	0.898	0.898	0.898	0.898	0.896	0.896	0.896	0.896	0.896
K	410	415	420	425	430	435	440	445	450	455
RMSE	0.895	0.895	0.895	0.894	0.894	0.894	0.894	0.893	0.893	0.893
K	460	465	470	475	480	485	490	495	500	505
RMSE	0.893	0.893	0.893	0.893	0.893	0.892	0.893	0.893	0.892	0.892
K	510	515	520	525	530	535	540	545	550	555
RMSE	0.892	0.892	0.891	0.891	0.891	0.891	0.890	0.890	0.890	0.890
K	560	565	570	575	580	585	590	595	600	605
RMSE	0.890	0.890	0.890	0.890	0.890	0.890	0.889	0.889	0.889	0.889
K	610	615	620	625	630	635	640	645	650	655
RMSE	0.888	0.888	0.888	0.888	0.888	0.888	0.888	0.888	0.888	0.888
K	660	665	670	675	680	685	690	695		
RMSE	0.888	0.888	0.888	0.888	0.888	0.888	0.887	0.888		

TABLE 4.4 – Données associées au graphe 4.11 (RMSE)

Comme nous pouvons le constater sur le graphe ci-dessus, la qualité de la prédiction évolue selon le nombre de voisins choisis. En prenant moins de 300 voisins, c'est à dire moins du tiers de l'ensemble de la communauté, la qualité semble diminuer.

Pour MAE, la meilleure valeur obtenu est de : 0.74

#### Apport du traitement des valeurs manquantes sur la technique précédente

Une importante caractéristique de ce data-set est que celui-ci contient un très grand nombre de valeurs manquantes : plus de 90%.

On pourrait supposer qu'en raison de grand taux de valeurs manquantes, leurs traitement pourrait biaiser les données et donc les résultats. Nous allons cependant essayer de les traiter et voir si cela améliore la qualité de la recommandation.

**User average** Nous avons remplacé les valeurs manquantes de chaque individu par la moyenne des ses évaluations :

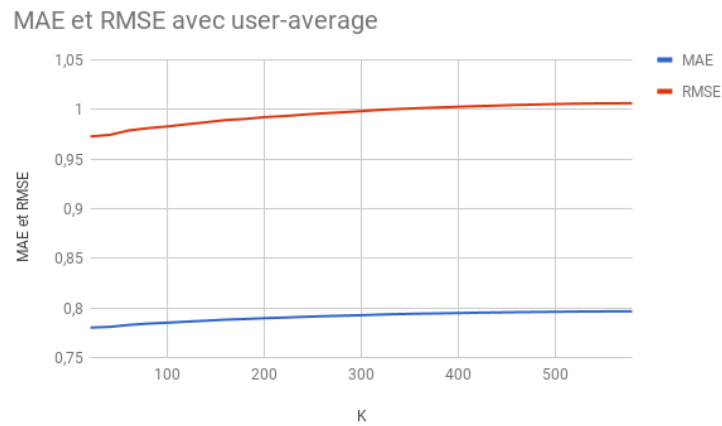


FIGURE 4.12 – Variation des résultats selon un filtrage classique basé sur KNN en fonction du paramètre k avec un traitement des valeurs manquantes en utilisant "user average"

K	20	40	60	80	100	120	140	160	180	200
MAE	0.780	0.780	0.782	0.784	0.785	0.786	0.787	0.788	0.788	0.789
K	220	240	260	280	300	320	340	360	380	400
MAE	0.790	0.791	0.791	0.792	0.792	0.793	0.793	0.794	0.794	0.794
K	420	440	460	480	500	520	540	560	580	
MAE	0.795	0.795	0.795	0.795	0.796	0.796	0.796	0.796	0.796	

TABLE 4.5 – Données associées au graphe 4.12 (RMSE)

K	20	40	60	80	100	120	140	160	180	200
RMSE	0.972	0.974	0.978	0.981	0.982	0.985	0.987	0.989	0.990	0.992
K	220	240	260	280	300	320	340	360	380	400
RMSE	0.993	0.994	0.996	0.997	0.998	0.999	1.000	1.001	1.002	1.002
K	420	440	460	480	500	520	540	560	580	
RMSE	1.003	1.003	1.004	1.004	1.005	1.005	1.006	1.006	1.006	

TABLE 4.6 – Données associées au graphe 4.12 (RMSE)

**item average** Nous avons remplacé les valeurs manquantes de chaque individu par la moyenne des évaluations sur l'item en question :

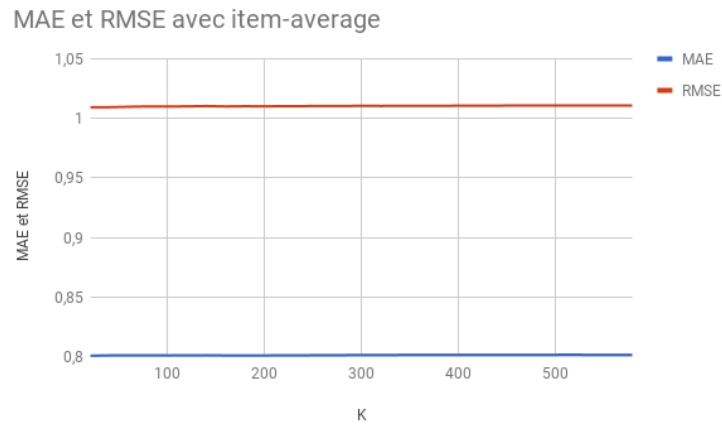


FIGURE 4.13 – Variation des résultats selon un filtrage classique basé sur KNN en fonction du paramètre k avec un traitement des valeurs manquantes en utilisant "item average"

K	20	40	60	80	100	120	140	160	180	200
MAE	0.8007	0.8010	0.8010	0.8011	0.8010	0.8011	0.8011	0.8009	0.8010	0.8009
K	220	240	260	280	300	320	340	360	380	400
MAE	0.8011	0.8011	0.8012	0.8012	0.8013	0.8013	0.8014	0.8014	0.8014	0.8014
K	420	440	460	480	500	520	540	560	580	
MAE	0.8014	0.8015	0.8015	0.8015	0.8015	0.8015	0.8015	0.8015	0.8015	

TABLE 4.7 – Données associées au graphe 4.13 (RMSE)

K	20	40	60	80	100	120	140	160	180	200
RMSE	1.0093	1.0093	1.0097	1.0100	1.0099	1.0101	1.0103	1.0100	1.0102	1.0101
K	220	240	260	280	300	320	340	360	380	400
RMSE	1.0102	1.0102	1.0104	1.0103	1.0105	1.0104	1.0104	1.0105	1.0105	1.0106
K	420	440	460	480	500	520	540	560	580	
RMSE	1.0106	1.0107	1.0107	1.0107	1.0107	1.0107	1.0107	1.0107	1.0107	

TABLE 4.8 – Données associées au graphe 4.13 (RMSE)

Comme on peut le remarquer, la qualité de la prédiction ne semble pas évoluer et à peu près quelque-soit la valeur de  $k$ . Ce phénomène est dû au fait que notre data-set présente un très grand taux de valeurs manquantes. Et le traitement des valeurs manquantes pour ce data-set biaise énormément les données et par conséquent les résultats.

### Vers un filtrage ItemItem

Jusque là, nous avons toujours adopté un filtrage de type user-user. Même si celui-ci est plus utilisé que le filtrage item-item, nous avons tout de même testé ce type de filtrage, toujours par rapport à la précédente technique (KNN).

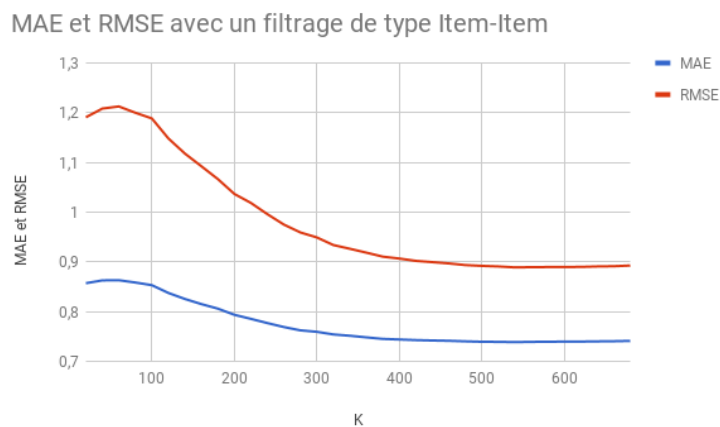


FIGURE 4.14 – Variation des résultats selon un filtrage classique basé sur KNN en fonction de paramètre  $k$  avec un traitement des valeurs manquantes en utilisant "item average"

K	20	40	60	80	100	120	140	160	180	200
MAE	0.857	0.862	0.863	0.858	0.853	0.837	0.825	0.815	0.806	0.793
K	220	240	260	280	300	320	340	360	380	400
MAE	0.785	0.777	0.769	0.762	0.759	0.754	0.751	0.748	0.745	0.743
K	420	440	460	480	500	520	540	560	580	600
MAE	0.742	0.741	0.741	0.740	0.739	0.739	0.738	0.739	0.739	0.739
K	620	640	660	680						
MAE	0.739	0.740	0.740	0.741						

TABLE 4.9 – Données associées au graphe 4.14 (MAE)

K	20	40	60	80	100	120	140	160	180	200
RMSE	1.190	1.208	1.212	1.199	1.188	1.148	1.117	1.092	1.066	1.036
K	220	240	260	280	300	320	340	360	380	400
RMSE	1.018	0.996	0.975	0.959	0.949	0.933	0.926	0.918	0.910	0.906
K	420	440	460	480	500	520	540	560	580	600
RMSE	0.902	0.899	0.896	0.893	0.892	0.890	0.888	0.889	0.889	0.889
K	620	640	660	680						
RMSE	0.889	0.890	0.891	0.892						

TABLE 4.10 – Données associées au graphe 4.14 (RMSE)

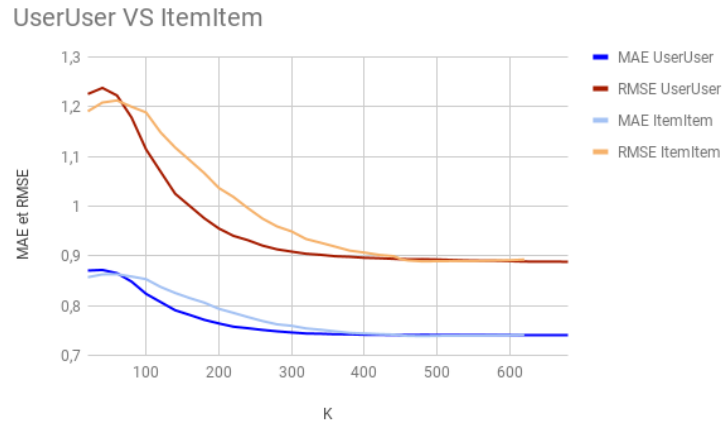


FIGURE 4.15 – UserUser vs ItemItem

Le graphe de la figure 4.15 illustre la différence entre un filtrage de type User-User et un filtrage de type Item-Item sur le jeu de données MovieLens 100K. On remarque qu'il y a une très légère différence entre les deux approches en terme de

qualité de prédiction. Cependant, la plupart des systèmes de recommandation basés sur un filtrage collaboratif se basent sur un filtrage de type User-User. Nous nous baserons également sur un filtrage collaboratif de type User-User même celui-ci ne semble pas donner de meilleurs résultats pour notre jeu de données.

### Apport du clustering sur le filtrage collaboratif

Comme nous l'avons expliqué lors du chapitre précédent, notre technique s'appuie sur un clustering d'individus. Nous testerons ainsi les performances d'un filtrage collaboratif avec un clustering d'individus en variant le nombre de cluster. Nous avons appliqué l'algorithme des k-medoids pour le processus de clustering.

Pour ce test, le paramètre indiquant la taille minimale d'un cluster sera fixé à 3, c'est à dire que tout cluster contenant moins de 3 éléments ne sera plus considéré.

#### Variation du nombre initial de clusters

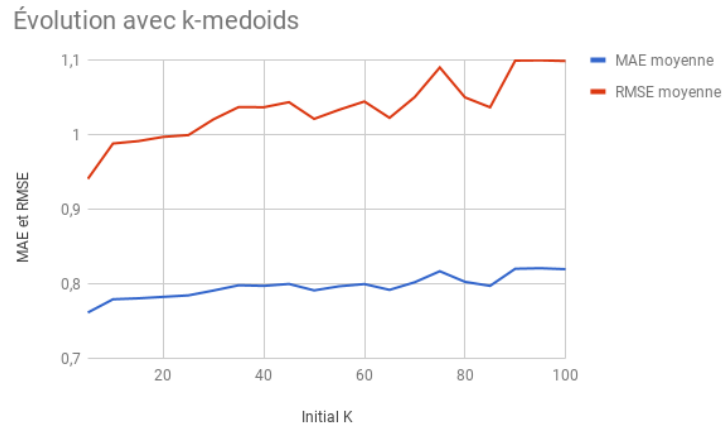


FIGURE 4.16 – Évolution de la performance du système selon le nombre de clusters indiqué initialement (k-medoids)

K	5	10	15	20	25	30	35	40	45	50
MAE	0.761	0.779	0.780	0.782	0.784	0.790	0.797	0.797	0.799	0.791
RMSE	0.940	0.988	0.991	0.996	0.999	1.020	1.036	1.036	1.043	1.021
K	55	60	65	70	75	80	85	90	95	100
MAE	0.796	0.799	0.791	0.802	0.816	0.802	0.797	0.820	0.820	0.819
RMSE	1.033	1.044	1.022	1.050	1.089	1.049	1.036	1.099	1.099	1.098

TABLE 4.11 – Données associées au graphe 4.16 (MAE & RMSE)

### Variation du nombre de clusters en sortie

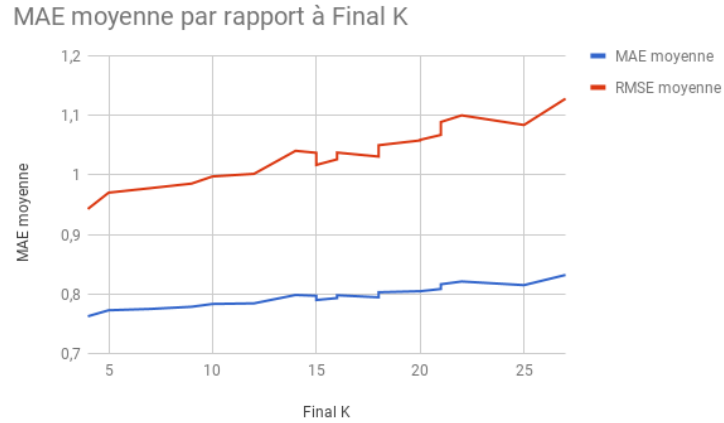


FIGURE 4.17 – Évolution de la performance du système selon le nombre de clusters obtenus en sortie (k-medoids)

K	5	4	7	9	10	12	14	15	15	16
MAE	0.772	0.762	0.774	0.778	0.783	0.784	0.798	0.797	0.789	0.793
RMSE	0.970	0.942	0.977	0.985	0.997	1.001	1.040	1.037	1.016	1.026
K	16	18	18	20	20	21	21	22	25	27
MAE	0.797	0.794	0.802	0.804	0.804	0.808	0.816	0.821	0.814	0.831
RMSE	1.037	1.030	1.049	1.057	1.058	1.067	1.088	1.099	1.083	1.128

TABLE 4.12 – Données associées au graphe 4.17 (MAE & RMSE)

On remarque que la qualité du système est notamment proportionnelle au nombre de clusters en sortie. Un grand nombre de clusters semble conduire à de mauvais résultats.

## Apport de l'optimisation sur le clustering

Ce traitement consiste à améliorer la qualité du clustering effectué sur notre ensemble d'individus en appliquant la méta-heuristique des colonies de fourmis.

**Le nombre de clusters** Comme nous l'avons vu précédemment, le clustering effectué avec l'algorithme des k-medoids montre très bien qu'un grand nombre de clusters mène à des résultats médiocres. Nous avons ainsi pris en compte cette observation et avons donc décidé de ne pas fixer le paramètre  $k$  (nombre de clusters) à des valeurs trop élevées, ce qui nous permettra d'aboutir à de meilleurs résultats.

Cependant, cette déduction ne reste valable que pour le data-set utilisé, et peut s'avérer fausse dans le cas d'une application sur un jeu de données différent.

**Le nombre d'itérations maximal** Il s'agit du critère d'arrêt de notre algorithme. En effet, le processus prendra fin une fois le nombre d'itérations maximal indiqué sera atteint. On peut supposer qu'un grand nombre d'itérations permettrait d'arriver à de meilleurs résultats. Nous avons donc effectué de nombreuses expérimentations en faisant évoluer ce paramètre afin de vérifier cette supposition.

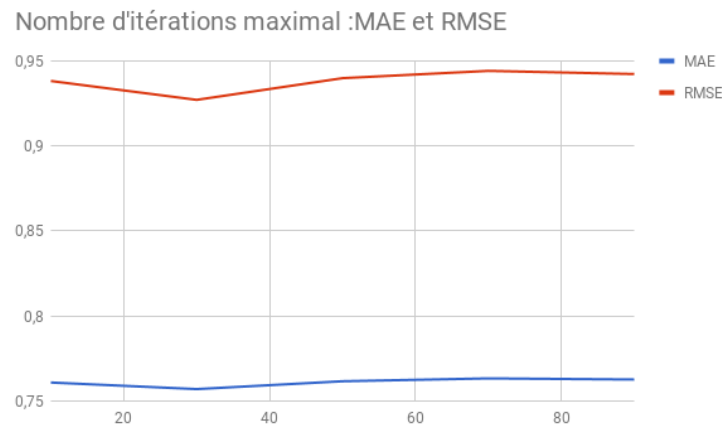


FIGURE 4.18 – Performances selon le paramètre *maxIter* de notre algorithme d'optimisation



K	10	30	50	70	90
MAE	0.760	0.757	0.761	0.763	0.762
RMSE	0.938	0.927	0.939	0.944	0.942

TABLE 4.13 – Données associées au graphe 4.18 (MAE & RMSE)

La figure 4.18 semble indiquer que la qualité des résultats ne s’améliore pas avec l’augmentation du nombre d’itérations.

**Remarque** Il a été également remarqué que la qualité du clustering n’est pas toujours proportionnelle à la qualité de la prédiction. Ainsi, un bon clustering n’est pas forcément synonyme d’un bon regroupement d’utilisateurs, notamment lorsque le nombre d’individus reste relativement petit (comme c’est le cas pour le data-set que nous utilisons).

**k-medoids vs k-medoids optimisé** Sur différents tests effectués, nous avons comparé la différence de performance après l’application de l’algorithme classique des k-medoids, et après l’application d’un clustering optimisé.

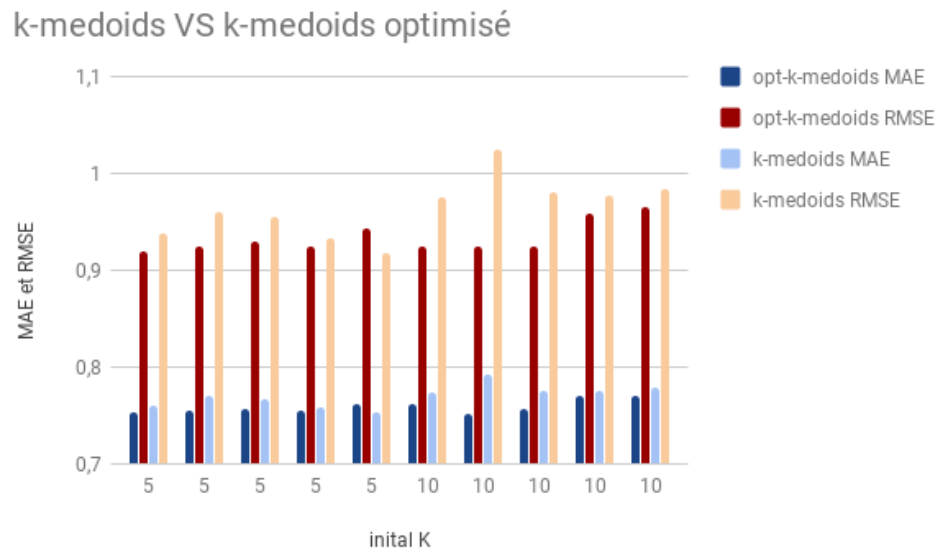


FIGURE 4.19 – k-medoids VS k-medoids optimisé

K initiaux	5	5	5	5	5
MAE (opt-k-medoids)	0,753935	0,75472	0,756675	0,75483	0,762075
RMSE (opt-k-medoids)	0,9199705	0,924168	0,9298125	0,924234	0,9423085
MAE (k-medoids)	0,759815	0,7694	0,766715	0,758525	0,752685
RMSE (k-medoids)	0,9379885	0,959409	0,9544615	0,9337475	0,9176325
K initiaux	10	10	10	10	10
MAE (opt-k-medoids)	0,761	0,752008	0,75718	0,77095	0,770195
RMSE (opt-k-medoids)	0,924074	0,924074	0,924074	0,957759	0,9645985
MAE (k-medoids)	0,774195	0,79201	0,77611	0,775305	0,778385
RMSE (k-medoids)	0,9747975	1,025198	0,979704	0,9763845	0,9840425

TABLE 4.14 – Données associées au graphe 4.19 (MAE & RMSE)

La figure ci-dessus illustre la différence de performance observé après l'application d'un clustering classique et après l'application d'un clustering optimisé. On remarque une légère amélioration des performances lorsqu'on a recourt à un clustering optimisé.

Appliqué sur des jeux de données plus volumineux, cette différence pourrait être encore plus remarquable.

### Comparaison avec l'état de l'art

En 2011, Jesus Bobadilla & all publie un article [30] où il propose une approche qui permettrait d'améliorer les performances des systèmes de recommandations basés sur le filtrage collaboratif en utilisant les algorithmes génétiques.

Ce travail, que nous avons réussi à reproduire et à implémenter, sera notre principal point de comparaison.

**Analyse et évaluation de l'algorithme** Étant donnée que cette approche est basée sur KNN, elle prend ainsi un paramètre  $k$  en entrée. Nous avons donc fait varier la valeur de ce paramètre en effectuant différents tests.

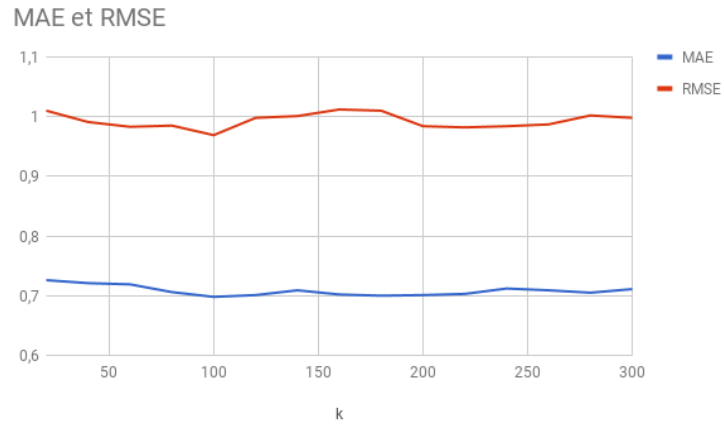


FIGURE 4.20 – Technique de j. Bobadilla [30]

K	20	40	60	80	100	120	140	160
MAE	0.726	0.721	0.719	0.706	0.698	0.701	0.709	0.707
RMSE	1.01	0.991	0.993	0.995	0.989	0.998	1.001	1.012
K	180	200	220	240	260	280	300	
MAE	0.7	0.701	0.703	0.712	0.709	0.705	0.711	
RMSE	1.01	0.994	0.992	0.994	0.997	1.002	0.998	

TABLE 4.15 – Tableau des données associées au graphe 4.20

Comme le montre le graphe obtenu, cette technique semble peu sensible au paramètre  $k$ , et offre de très bon résultats :

- best mae = 0.698
- best rmse = 0.989

### Tableau comparatif

Techniques	Métriques	
	MAE	RMSE
Filtrage classique	0.741	0.890
KNN	0.74	0.891
Approche décrite dans [30]	0.698	0.989
k-medoids(clustering)	0.752	0.917
Clustering optimisé (Notre approche)	0.748	0.905

TABLE 4.16 – Tableau comparatif des différentes techniques de FC testées

Ce tableau résume l'ensemble de ce qui a été fait dans la partie "expérimentations". En effet, celui-ci indique les performances de chaque technique implémentée et évaluée. Ces chiffres représentent les meilleures performances observées chez chacun de ces algorithmes.

Sur le data-set MovieLens 100k, notre approche n'a pas réellement offert des résultats meilleurs que ceux apportés par les algorithmes de filtrage collaboratif classiques. Par ailleurs, la comparaison avec la technique proposée par J. Bobadilla montre que celle-ci n'a pas tout à fait apporté des résultats meilleurs que ceux apportés par notre algorithme. Certes, Cette première a abouti à une *erreur moyenne absolue* inférieure, donc meilleure. Cependant, l'erreur quadratique moyenne observée est bien plus élevée.

## 4.6 Conclusion

Dans ce chapitre que nous clôturons, nous avons pu tester différents algorithmes classiques de recommandation existant dans la littérature sur le data-set MovieLens 100k. Nous avons également évalué l'algorithme que nous avons proposé sur ce même data-set.

En effet, ces expérimentations nous permettent d'émettre certaines conclusions sur notre approche. Bien que celle-ci ne semble pas apporter une nette amélioration en terme de qualité, elle offre tout de même des résultats très satisfaisants, qui pourraient très bien être meilleurs sur des jeux de données autres que celui utilisé dans notre travail.

# Conclusion

Les systèmes de recommandation sont aujourd’hui plus présents que jamais sur le web. Leur apport dans le e-commerce, l’e-service et les réseaux sociaux entre autres est indéniable, où ils répondent de fort belle manière aux attentes des prestataires de services, des nombreuses plates-formes présentes sur le net et des utilisateurs.

En effet, la recommandation personnalisée devient un enjeu de taille notamment pour les géants du e-commerce, si bien que ceux-ci sont constamment amenés à booster les performances de leurs systèmes de recommandation en vue d’améliorer la qualité de leurs services. Il a été révélé que 30 des ventes sur Amazon sont réalisées grâce à la recommandation [11].

Nous nous sommes intéressés dans le cadre de ce travail à la proposition d’une amélioration au filtrage collaboratif, l’un des algorithmes les plus utilisés dans les systèmes de recommandation.

Nous avons proposé une approche de recommandation par filtrage collaboratif basé sur une classification non supervisée. L’identification du voisinage des utilisateurs s’effectue grâce à l’utilisation de l’algorithme de clustering des k-medoids. Nous avons également voulu optimiser ce processus en considérant la méta-heuristique des colonies de fourmis. Dans notre cas de figure, l’optimisation du clustering reviendrait à trouver les medoids (ou centroids) de départ qui mèneront vers un partitionnement de qualité.

Nous avons implémenté et évalué les différents algorithmes de notre approche avec la base MovieLens100K incluant 100,000 évaluations (1-5) effectuées par 943 utilisateurs sur 1682 films. Les résultats obtenus montrent que la classification optimisée a amélioré le processus de construction des groupes d’utilisateurs ayant des similarités similaires. Par contre la qualité de la recommandation n’a pas été nettement améliorée comparativement aux performances du FC clas-

sique. Ceci serait dû selon notre interprétation au nombre limité de données de la base MovienLens 100K que nous avons considéré et nous pensons que pour voir l'apport de la classification optimisée sur le FC il faudrait considérer une base plus volumineuse telle que MovieLens 1M. Les résultats obtenus restent prometteurs, ce qui nous motive à creuser davantage cette problématique, en considérant les perspectives suivantes :

- Une évaluation plus approfondie de notre approche avec la base MovieLens1M, mais aussi avec d'autres bases dans d'autres domaines telle que le contexte social (exemple la base RED : Rich Epinions Dataset).
- Considérer d'autres techniques de clustering telles que SVM (Support Machine Vector).
- Adapter d'autres méta-heuristiques au problème d'optimisation de classification.
- Construction de systèmes basés sur cet approche, qui pourraient être exploitées dans le secteur socio-économique de notre pays.

# Bibliographie

- [1] <https://www.solver.com/xlminer/help/hierarchical-clustering-intro>.
- [2] <http://pro.arcgis.com/fr/pro-app/tool-reference/spatial-statistics/how-density-based-clustering-works.htm>.
- [3] [http://igm.univ-mlv.fr/~dr/XPOSE2013/tleroux\\_genetic\\_algorithm/fonctionnement.html](http://igm.univ-mlv.fr/~dr/XPOSE2013/tleroux_genetic_algorithm/fonctionnement.html).
- [4] <https://pandas.pydata.org/pandas-docs/stable/>.
- [5] Algorithme de colonies de fourmis. <http://www.apprendre-en-ligne.net/info/algo/fourmis.html>.
- [6] support vector machine (svm). <https://whatistechtarget.com/definition/support-vector-machine-SVM>.
- [7] Gediminas Adomavicius and Alexander Tuzhilin. To-ward the next generation of recommender systems : A survey of the state-of-the-art and possible extensions. *iee trans.knowl. data eng.* 2005.
- [8] M. S. ALDENDERFER and R. K. BLASHFIELD. Cluster analysis. 1984.
- [9] Souquet Amédée and Radet Francois-Gérard. *ALGORITHMES GENE-TIQUES*. PhD thesis.
- [10] COSTANZO Andrea, LUONG Thé Van, and MARILL Guillaume. *Optimisation par colonies de fourmis*, mai 2006.
- [11] Thierry Bedoucha. les moteurs de recommandation. <https://fr.slideshare.net/webcampday/thierry-bedoucha-les-moteurs-de-recommandation>, mai 2015.
- [12] Younes Benzaki. Naive bayes classifier pour machine learning. <https://mrhint.fr/naive-bayes-classifier>.
- [13] Daniel Billsus and Michael J. Pazzani. User modeling for adaptive news access, 2000.
- [14] Robin Burke. Hybrid recommender systems : Survey and experiments. 2002.

- [15] Robin Burke. Hybrid web recommender systems. 2007.
- [16] Benjamin Carro. <https://www.mediego.com/fr/blog/netflix-success-story-basee-sur-algorithmes-de-recommandation/>, 2010.
- [17] Sylvain Castagnos and Anne Boyer. Modeling preferences in a distributed recommender system. 2007.
- [18] Nathalie Denos Catherine Berrut. Assistance intelligente à la recherche d'informations. 2003.
- [19] Y. H. Chien and E. I. George. A bayesian model for collaborative filtering. 1999.
- [20] Brian M. Oki David Goldberg, David Nichols and Douglas Terry. Using collaborative filtering to weave an information tapestry. commun. acm . Dec 1992.
- [21] MARCO DORIGO. L'intelligence en essaim. <http://www.prixquinquennaux.be/docs/MarcoDorigo.pdf>.
- [22] ecommerceresult. [www.ecommerceresult.com/fr/logiciel-de-recommandation/](http://www.ecommerceresult.com/fr/logiciel-de-recommandation/).
- [23] Jin-Kao Hao et Christine Solnon. Méta-heuristiques et intelligence artificielle.
- [24] M. Clerc et P. Siarry. Une nouvelle métaheuristique pour l'optimisation difficile : la méthode des essaims particulaires. Oct 2004.
- [25] Bracha Shapira Francesco Ricci, Lior Rokach. *Introduction to recommender systems handbook*. Springer, 2011.
- [26] David Heckerman Guy Shani and Ronen I. Brafman. An mdp-based recommender system. Dec 2005.
- [27] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining : Concepts and techniques*. Morgan Kaufmann, 2012.
- [28] J. ; Borchers A. et Riedl J Herlocker, J. ; Konstan. . an algorithmic framework for performing collaborative filtering. 1999.
- [29] A. K. JAIN and R. C. DUBES. Algorithms for clustering data. 1988.
- [30] Antonio Hernando Javier Alcalá Jesus Bobadilla, Fernando Ortega. Improving collaborative filtering recommender system results and performance using genetic algorithms. 2011.
- [31] D. Heckerman J.S. Breese and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. 1998.
- [32] A. Kohrs and B Meriardo. Clustering for collaborative filtering applications. 1999.
- [33] Ken Lang. Newsweeder : Learning to filter netnews. *CiteSeerX*, 1995.
- [34] Pasquale Lops, Marco De Gemmis, Giovanni Semeraro, Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. Chapter 3 content-based recommender systems : State of the art and trends.



- [35] Ralf Klamma Matthias Jarke Manh Cuong Pham, Yiwei Cao. A clustering approach for collaborative filtering recommendation using social network analysis. Oct 2010.
- [36] Jiirg Sander Xiaowei Xu Martin Ester, Hans-Peter Kriegel. A density-based algorithm for discovering clusters in large spatial databases with noise. 1996.
- [37] MovieLens. <https://www.mediego.com/fr/blog/netflix-success-story-basee-sur-algorithmes-de-recommandation/>, 2010.
- [38] M. STEINBACH P.-N. TAN and V. KUMAR. Introduction to data mining. 2005.
- [39] Michael Pazzani and Daniel Billsus. Learning and revising user profiles : The identification of interesting web sites. *Springer*, 1997.
- [40] Romain Picot-Clemente. Les systèmes de recommandation. <https://fr.slideshare.net/rpicotcl/presentation-decide-21-072015>, juil 2015.
- [41] Ricco Rakotomalala. Filtrage collaboratif et système de recommandations. <https://eric.univ-lyon2.fr/~ricco/cours/slides/WM.B%20-%20Filtrage%20Collaboratif%20-%20Recommandation.pdf>. univ Lyon2.
- [42] Andriy Mnih Ruslan Salakhutdinov and Geoffrey Hinton. Restricted boltzmann machines for collaborative filtering. 2007.
- [43] G. Salton. *Automatic Text Processing*. Addison-Wesley, 1989.
- [44] Kelvin Salton. How dbscan works and why should we use it ? <https://towardsdatascience.com/how-dbscan-works-and-why-should-i-use-it-443b4a191c80>, 2017.
- [45] Frédéric Santos. *Arbres de décision*, Mar 2015.
- [46] Kinzel D. Schafer, M. and Winkler. continuous organization and specification of the lateral floor plate in zebrafish. 2007.
- [47] George Seif. The 5 clustering algorithms data scientists need to know. <https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68>, feb 2018. AI / Machine Learning Engineer.
- [48] Suman and Pinki Rani. A survey on sting and clique grid based clustering methods. May-June 2017.
- [49] Bing Pan Helene Hembrooke Thorsten Joachims, Laura Granka and Geri Gay. Accurately interpreting clickthrough data as implicit feedback. 2005.
- [50] Sonia Ben Ticha. *Recommandation personnalisée hybride*. PhD thesis, Université de Lorraine, 2015.
- [51] Lyle H. Ungar and Dean P. Foster. Clustering methods for collaborative filtering. 1998.
- [52] Thanh Trung Van. Utilisation de profils utilisateurs pour l'accès à une bibliothèque numérique. 2008. Ecole Nationale Supérieure des Mines de Saint-Etienne.

- [53] Wikipédia.
- [54] wikiversité. [https://fr.wikiversity.org/wiki/Python/Pr%C3%A9sentation\\_de\\_la\\_le%C3%A7on](https://fr.wikiversity.org/wiki/Python/Pr%C3%A9sentation_de_la_le%C3%A7on).
- [55] Philip Zigoris and Yi Zhang. Bayesian adaptive user profiling with explicit and implicit feedback. 2006.
- [56] Sami Äyrämö and Tommi Kärkkäinen. Introduction to partitioning-based clustering methods with a robust example. 2006.
- [57] Johann Dréo Alain Pétrowski Patrick Siarry Éric Taillard. *Métaheuristiques pour l'optimisation difficile*. Eyrolles, 2003.