



République Algérienne Démocratique et Populaire  
Université Abou Bakr Belkaid– Tlemcen  
Faculté Des Sciences  
Département d'Informatique

Mémoire de fin d'études

Pour l'obtention du diplôme de Master en Informatique  
Option SIC

*Thème*

**Application et exploration de l'approche  
de découpage des systèmes de  
recommandations pré-contextuelle**

**Réalisé par :**

- MESROUA NADJLA AMINA
- SEGHIRI DJIHANE

*Présenté le 27 Juin 2019 devant le jury composé de MM .*

- MAATALLAH H. (Président)
- BENMANSOUR F. (Examinatrice)
- EL YEBDRI Z. (Encadreur)



# Remerciement


*Nous tenons tout d'abord à remercier Dieu le tout puissant et miséricordieux, qui nous a donné la force et la patience d'accomplir ce modeste travail.*

*En second lieu, nous tenons à remercier notre encadreur Mme EL  
YESEDRA ZMEB. Pour son encadrement, ses conseils et son aide  
précieux et constant qu'elle nous a apporté tout au long de ce travail, ainsi  
que pour les remarques constructives qu'elle nous a donné lors de la rédaction  
de ce mémoire.*

*Nos vifs remerciements vont également aux membres de jury pour l'intérêt  
qu'ils ont porté à notre recherche en acceptant d'examiner notre travail et de  
l'enrichir par leurs propositions.*

*Nous adressons notre plus sincère remerciement à tous les enseignants de  
département d'informatique, qui par leur enseignement, ont contribué à notre  
formation durant tout notre cursus universitaire.*

*Enfin, nous tenons également à remercier toutes les personnes qui ont participé  
de près ou de loin à la réalisation de ce travail.*





# Dédicace

*Je dédie ce modeste travail à celle qui m'a donné la vie. Qui s'est sacrifiée pour mon bonheur et ma réussite, à ma mère le symbole de tendresse.*

*À mon père, école de mon enfance, qui a été mon ombre durant toutes mes années d'études, et qui a veillé tout au long de ma vie à m'encourager, à me donner de l'aide et à me protéger, que dieu le garde et le protège.*

*Aucune dédicace ne saurait exprimer ma grande admiration, ma considération et ma sincère affection pour vous deux.*

*À ma sœur unique FEDOUA avec laquelle je partage une immense complicité à travers laquelle une très forte affection est née entre nous et restera à jamais, rien au monde ne pourra nous séparer, et à son fils adoré WASSIM que dieu le garde pour elle, ainsi que son mari MOHAMMED.*

*À mon frère adoré Faysal ainsi que sa femme FU ELLA et leur petite princesse qui a vu le jour récemment. Vous vous êtes toujours préoccupée de moi en m'octroyant un soutien morale inestimable et apaisé. Vous m'avez constamment annoncé de bonnes nouvelles.*

*Merci pour tout*

*Que Dieu le Tout puissant vous comble de Sa grâce et de Sa protection.*

*À toi NEDJLA, qui as été ma plus belle rencontre scientifique et amicale.*

**SEGHRA DJAHANE**



# Dédicace

*Je dédie ce modeste travail à celle qui m'a donné la vie. Qui s'est sacrifiée pour mon bonheur et ma réussite, à ma mère le symbole de tendresse.*

*À mon père, école de mon enfance, qui a été mon ombre durant toutes mes années d'études, et qui a veillé tout au long de ma vie à m'encourager, à me donner de l'aide et à me protéger, que dieu le garde et le protège.*

*Aucune dédicace ne saurait exprimer ma grande admiration, ma considération et ma sincère affection pour vous deux.*

*À ma sœur unique IKRAM FATMA ZOÛRA avec laquelle je partage une immense complicité à travers laquelle une très forte affection est née entre nous et restera à jamais, rien au monde ne pourra nous séparer.*

*À mon beau-frère adoré SIDI MOHAMMED, et ma meilleure amie NESRME.*

*Par vos mots apaisés, vos conseils inestimables et vos encouragements, vous avez toujours su me pousser à croire et aller de l'avant. Aujourd'hui aucun mot ne peut exprimer ma reconnaissance.*

*Que Dieu le Tout puissant vous comble de Sa grâce et de Sa protection ainsi que toute votre famille*

*À toi DJIHANE, qui as été ma plus belle rencontre scientifique et amicale.*

**MESROUA NEDJLA AMINA**

# Sommaire

Liste des figures .....	7
Liste des tableaux .....	8
INTRODUCTION GENERALE.....	9
Introduction générale.....	10
Chapitre1.Les Systèmes de recommandations classiques.....	13
1.1 Introduction .....	14
1.2. Historique .....	14
1.3. Définition des systèmes de recommandations.....	15
1.4. Objectif des systèmes de recommandation.....	16
1.5. Les étapes principales de la recommandation .....	16
1.5.1. Collecte d'information .....	17
1.5.2. Mise en œuvre d'une matrice utilisateur d'informations récoltées .....	18
1.5.3. Extraction de la liste de recommandations.....	19
1.6. Les principales approches de filtrage .....	19
1.6.1. Filtrage basé sur le contenu (content based filtering) .....	19
1.6.2. Filtrage collaboratif (collaborative filtering).....	20
1.6.3. Filtrage hybride (hybrid filtering) .....	22
1.7. Avantages et inconvénients des systèmes de recommandations .....	24
1.8. Evaluation des Systèmes de recommandation.....	26
1.8.1. Évaluation online.....	26
1.8.2. Evaluation offline .....	27
1.9. Conclusion .....	28
Chapitre2.Les systèmes de recommandation sensible au contexte .....	29
2.1. Introduction .....	30
2.2 Définition du contexte .....	30
2.3 Sources d'informations contextuelles.....	31
2.4 Caractéristiques du Contexte .....	32
2.5 Dimension du Contexte .....	32
2.6 Systèmes sensibles au contexte .....	33
2.7 . Approches d'intégration du contexte dans les systèmes de recommandation .....	34
2.7.1 Approche de Pré-filtrage Contextuel (contextual pre-filtering).....	34
2.7.2. Approche de Post-filtrage Contextuel (contextual post-filtering).....	35
2.7.3. Approche de la Modélisation Contextuelle (contextual modeling).....	36
2.8. Exemples de Système de Recommandation sensibles au contexte .....	37
2.8.1. Micro-profiling.....	37
2.8.2. Amazon .....	38
2.9. Splitting approche du pré-filtrage contextuel .....	38
2.10. Conclusion .....	39
Chapitre3.Implémentation de l'approche de découpage .....	40
3.1. Introduction .....	41
3.2. Terminologies utilisées.....	41
3.3. Description de l'approche de découpage.....	42
3.3.1. Algorithmes de découpage.....	42
3.3.1.1. Découpage par item.....	42



3.3.1.2.	Découpage par utilisateur .....	43
3.3.1.3.	Découpage par item&utilisateur .....	44
3.3.2.	Critères d'impuretés .....	44
3.4.	Implémentation de l'approche de découpage .....	45
3.4.1.	Processus de découpage .....	46
3.4.2.	Calcul taux de similarité .....	47
3.4.3.	Calcul de prédiction .....	47
3.4.4.	Top N Recommandation .....	47
3.5.	Evaluation .....	47
3.5.1.	Description du Dataset .....	48
3.5.2.	Méthode d'évaluation .....	48
3.5.3.	Résultat .....	48
3.6.	Conclusion .....	49
Chapitre4.	Utilisation de la Bibliothèque CarsKit .....	50
4.1.	Introduction .....	51
4.2.	Conception .....	51
4.3.	Algorithmes .....	51
4.3.1.	Algorithmes de transformations .....	52
4.3.2.	Algorithmes d'adaptations .....	52
4.4.	Evaluation .....	52
4.5.	Guide d'utilisation .....	53
4.5.1.	Téléchargement de fichier CARSKit .....	53
4.5.2.	Format de donnée .....	54
4.5.3.	Préparation de donnée .....	55
4.5.4.	Configuration .....	55
4.5.	Conclusion .....	58
Conclusion générale	.....	60
Références Bibliographiques	.....	62
Liste des Abréviation	.....	68
Résumé	.....	68

# *Liste des figures*

Figure 1 : Les étapes d'un système de recommandation .....	17
Figure 2 : Un système de recommandation basé sur le contenu .....	20
Figure 3 : Un système de recommandation collaboratif .....	21
Figure 4 : Le système de recommandation hybride .....	22
Figure 5 : Conception d'hybridation monolithique.....	23
Figure 6 : Conception d'hybridation parallèle .....	24
Figure 7 : Conception d'hybridation tubulaire.....	24
Figure 8 : Approche Pré-filtrage Contextuel .....	35
Figure 9 : Approche Post-filtrage contextuel .....	36
Figure 10 : Approche Modélisation contextuelle .....	37
Figure 11 : Barre d'outils du site <a href="http://www.amazon.com">www.amazon.com</a> .....	38
Figure 12 Architecture de l'approche de découpage.....	46
Figure 13 : Résultat d'évaluation .....	48
Figure 14 : Architecture de CARSKIT .....	51
Figure 15 : format de donnée .....	54
Figure 16 : format binaire .....	54
Figure 17 : Résultat .....	57

# *Liste des tableaux*

Tableau 1 : Exemple de matrice utilisateur .....	19
Tableau 2 : Les avantages et les inconvénients des techniques de recommandation.....	26
Tableau 3 : Exemple de jeu de donnée.....	41
Tableau 4 : Matrice de notation résultat découpage par item.....	43
Tableau 5 : Matrice de notation résultat découpage par item.....	43
Tableau 6 : Matrice de notation résultat découpage par item&utilisateur .....	44



*INTRODUCTION*  
*GENERALE*

## **Introduction générale**

Compte tenu, de la très grande masse d'informations aujourd'hui disponible sur Internet et le besoin en communication, en échange d'idées et en partage d'informations, les besoins de l'utilisateur sont devenus difficiles, non pas dans le sens de disposer de l'information, mais de trouver l'information pertinente au bon moment.

Donc il est devenu primordial de concevoir des mécanismes qui permettent aux utilisateurs d'accéder à ce qui les intéresse le plus rapidement possible. Delà les systèmes de recommandations se sont apparus.

Durant les 20 dernières années, les systèmes de recommandations (SRs) se sont répandus dans le domaine de la recherche en information, pour résoudre le problème des surcharges d'informations en recommandant des items (qui peuvent être un film, un livre ou un endroit à visiter ...) appropriés à un utilisateur .

Les SRs utilisent des profils représentant des préférences des utilisateurs pour calculer des recommandations. Ce calcul se fait par la prédiction des notes (ratings) qu'un utilisateur est susceptible d'attribuer aux items.

Les approches de recommandation se basent, généralement, sur le filtrage basé sur le contenu (FBC) qui recommande à l'utilisateur des items correspondant à son profil, le filtrage collaboratif (FC) qui consiste à recommander à l'utilisateur des items choisis (ou bien notés) par les utilisateurs similaires ou sur une combinaison de ces deux approches appelés filtrage hybride.

### **Problématique et objectif**

Les SRs traditionnels modélisent le degré d'utilité d'un item à un utilisateur qu'il ne l'a pas déjà évalué par rapport aux items notés dans le passé.

Cependant, ces systèmes ignorent que ces notes attribuées peuvent être différentes en allant d'un contexte à un autre. Par exemple, l'endroit où se trouve l'utilisateur (à la maison ou au travail), le temps (le matin ou le soir), la saison, l'état d'esprit peuvent influencer sur les votes des utilisateurs, et aussi améliorer l'exactitude des recommandations.

Les systèmes de recommandation sensibles au contexte (les CARS : ContextAware Recommender Systems) apportent une solution à ce problème. Sachant que récemment, les entreprises ont commencé à intégrer certaines informations contextuelles dans leurs moteurs de recommandation.

Trois approches principales dans lequel les informations contextuelles peuvent être incorporées dans les systèmes de recommandation à savoir : Pré-filtrage Contextuel (contextual pre-filtering), Post-filtrage Contextuel (contextual post-filtering), Modélisation Contextuelle (contextual modeling).

Dans notre travail, nous nous intéressons plus particulièrement à l'approche de découpage qui appartient à l'approche de pré-filtrage contextuel. Les algorithmes de découpage sont considérés comme les plus efficaces et les plus populaires parmi les autres algorithmes de recommandation contextuelle. Trois types d'algorithmes sont proposés, dont nous citons découpage par item, découpage par utilisateur et découpage par item&utilisateur.

L'objectif de notre travail est d'appliquer ces algorithmes sur des datasets réels.

Pour cela, nous avons divisé notre travail en deux parties. La première partie consiste à implémenter l'un des algorithmes de découpage à savoir découpage par item. La deuxième partie consiste à explorer la bibliothèque CARSKits basée sur Java spécialement conçue pour la recommandation sensible au contexte, et qui permet d'évaluer l'ensemble des algorithmes de recommandation sensible au contexte pour la partie expérimentation des travaux de recherche dans le domaine. Le but de comparer les travaux proposés avec les travaux existant dans le domaine.

## **Organisation**

Ce mémoire comporte quatre chapitres qui se répartissent comme suit :

Le premier chapitre, présente une vue générale sur les systèmes de recommandation cette vue générale présente l'histoire des systèmes de recommandation, suivie de leur définition, puis les différentes approches de recommandations qui existent, ainsi que leurs avantages et inconvénients.

Dans le deuxième chapitre, nous commençons par définir la notion de contexte, par la suite nous décrivons les différentes approches d'intégration du contexte. Nous terminons par définir l'approche de découpage.

Le troisième chapitre nous avons exploré l'approche de découpage en décrivant la méthode d'évaluation ainsi que la description de notre dataset

Et le dernier chapitre, nous avons détaillé les différentes étapes de l'utilisation de la bibliothèque de CARSKit

# Chapitre 1

*Les Systèmes de  
recommandations  
classiques*

## 1.1 Introduction

Avec la croissance de popularité des applications web et l'explosion de l'utilisation de l'Internet pendant les années '90 nous avons senti le besoin de filtrer cette énorme quantité d'informations.

Donc il est nécessaire de disposer d'un système qui peut recommander des bons produits en tenant compte de certains paramètres. Il s'agit là du système de recommandation qui est devenu plus populaire durant ces 10 à 20 dernières années.

Un système de recommandation fournit à des utilisateurs des suggestions qui répondent à leurs besoins et préférences informationnels.

Les applications de recommandation peuvent être trouvées dans une grande variété d'industries, entreprises, service financier, musique / radio en ligne, TV et vidéos, les publications en ligne, et d'innombrables autres.

Dans ce chapitre, nous commençons par donner un historique des systèmes de recommandation puis nous définirons les systèmes de recommandation d'une manière globale, ainsi les différentes approches de recommandations et nous finirons par exploiter leurs avantages et inconvénients.

## 1.2. Historique

Les racines des systèmes de recommandation remontent aux travaux étendus dans les sciences cognitives, la théorie d'approximation, la recherche d'informations, la théorie de la prévoyance ont également des liens avec la science de la gestion et le marketing, dans la modélisation des choix du consommateur [1].

« Information Lens System » [28] peut être considérée comme le premier système de recommandation. À l'époque, l'approche la plus commune pour le problème de partage d'informations dans l'environnement de messagerie électronique était la liste de distributions basée sur les groupes d'intérêt. La première définition pour le filtrage a été donnée aussi par Malone : « Même si le terme a une connotation littérale de laisser les choses dehors (filtrage négatif : enlèvement), nous l'utilisons ici dans un sens plus général qui consiste à

Sélectionner les choses à partir d'un ensemble plus large de possibilités (filtrage positif : sélection) ».

Au cours des années 90, les systèmes de recommandation sont devenus un important domaine de recherche avec la publication des premiers articles dans le domaine du filtrage collaboratif. La littérature académique a introduit le terme de filtrage collaboratif par le système Tapestry [21]. Il a été développé en 1992 par le centre de recherche de "Xerox" aux Etats Unis et qui a permis aux utilisateurs de créer des requêtes permanentes, basées sur les annotations des utilisateurs.

À la fin des 1990, Les déploiements de la technique de recommandations commerciales ont commencé à émerger. Peut-être la plus célèbre application sur une large gamme de systèmes techniques de recommandation est Amazon.com, En suit, en 2006, Netflix a lancé Netflix Prize pour améliorer l'état de recommandation des films ; aussi Netflix a plus de 17.000 films dans sa sélection [32].

Aujourd'hui les systèmes de recommandation sont devenus très populaires et sont utilisés dans diverses applications Web.

### 1.3. Définition des systèmes de recommandations

Les systèmes de recommandation peuvent être définis de plusieurs façons, vue la diversité des classifications proposées pour ces systèmes qui peuvent se rapporter à différents types de données ou approches spécifiques, mais il existe une définition générale de Robin Burke [10] que nous utiliserons dans ce mémoire et qui les définit comme suit :

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

Les deux entités de base qui apparaissent dans tous les systèmes de recommandations sont l'utilisateur et l'item. L'«**usager**» est la personne qui utilise un système de recommandation, donne son opinion sur diverses items et reçoit les nouvelles recommandations du système. L'«**Item**» est le terme général utilisé pour désigner ce que le système recommande aux usagers.

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



- **Les estimations** : (également appelées les votes), expriment l'opinion des utilisateurs sur les articles (exemple : 1 mauvais à 5 excellent).
- **Les données démographiques** : se réfèrent à des informations telles que l'âge, le sexe, le pays et l'éducation des utilisateurs. Ce type de données est généralement difficile à obtenir et est normalement collecté explicitement.
- **Les données de contenu** : qui sont fondées sur une analyse textuelle des documents liés aux éléments évalués par l'utilisateur. Les caractéristiques extraites de cette analyse sont utilisées comme entrées dans l'algorithme de filtrage afin d'en déduire un profil d'utilisateur [30].

#### 1.4. Objectif des systèmes de recommandation

Un système de recommandation a pour objectif de fournir à un utilisateur des ressources pertinentes en fonction de ses préférences. Ce dernier voit ainsi réduit son temps de recherche mais reçoit également des suggestions de la part du système auxquelles il n'aurait pas spontanément prêtées attention. L'essor du Web et sa popularité ont notamment contribué à la mise en place de tels systèmes comme dans le domaine du e-commerce. Les systèmes de recommandation peuvent être vus initialement comme une réponse donnée aux utilisateurs ayant des difficultés à prendre une décision dans le cadre d'utilisation d'un système de recherche d'information « classique ».

#### 1.5. Les étapes principales de la recommandation

Un système de recommandation requiert généralement trois(03) étapes, et qui sont illustré dans la figure suivante :



**Figure 1 :** Les étapes d'un système de recommandation

### 1.5.1. Collecte d'information

Pour être pertinent, un système de recommandation doit pouvoir faire des prédictions sur les intérêts des utilisateurs. Il faut donc pouvoir collecter un certain nombre de données sur ceux-ci afin d'être capable de construire un profil pour chaque utilisateur. Une distinction peut être faite entre 2 formes de collecte de données :

- **Collecte de données explicites - Filtrage actif :** La collecte repose sur le fait que l'utilisateur indique explicitement au système ses intérêts.

**Exemple :** Demander à un utilisateur de commenter, taguer/étiqueter, noter, liker ou encore ajouter comme favoris des contenus (objets, articles...) qui l'intéressent. On utilise souvent une échelle de ratings allant de 1 étoile (je n'aime pas du tout) à 5 étoiles (j'aime beaucoup) qui sont ensuite transformées en valeurs numériques afin de pouvoir être utilisées par les algorithmes de recommandation.

**Avantage :** Capacité à reconstruire l'historique d'un individu et capacité à éviter d'agréger une information qui ne correspond pas à cet unique utilisateur (plusieurs personnes sur un même poste).

**Inconvénient :** Les informations recueillies peuvent contenir un biais dit de déclaration.

- **Collecte de données implicite - Filtrage passif** : Elle repose sur une observation et une analyse des comportements de l'utilisateur effectué de façon implicite dans l'application qui embarque le système de recommandation, le tout se fait en "arrière-plan" (en gros sans rien demander à l'utilisateur).

**Exemple :**

- Obtenir la liste des éléments que l'utilisateur a écoutés, regardés ou achetés en ligne ;
- Analyser la fréquence de consultation d'un contenu par un utilisateur, le temps passé sur une page ;
- Monitorer le comportement en ligne de l'utilisateur ;
- Analyser son réseau social.

**Avantage** : Aucune information n'est demandée aux utilisateurs, toutes les informations sont collectées automatiquement. Les données récupérées sont a priori justes et ne contiennent pas de biais de déclaration [26].

**Inconvénient** : Les données récupérées sont plus difficilement attribuables à un utilisateur et peuvent donc contenir des biais d'attribution (utilisation commune d'un même compte par plusieurs utilisateurs). Un utilisateur peut ne pas aimer certains livres qu'il a achetés, ou il peut l'avoir acheté pour quelqu'un d'autre [26]

### 1.5.2. Mise en œuvre d'une matrice utilisateur d'informations récoltées

La mise en œuvre d'une matrice appelée " matrice utilisateur " ou bien " modèle utilisateur " incluant les informations concernant les utilisateurs récoltées durant la précédente étape qu'est la collecte d'informations. On peut la représenter comme un tableau qui contient des données recueillies sur l'utilisateur associées aux produits disponibles sur le site web

	Item 1	Item 2	Item 3	Item 4	Item 5	...
User 1	😊			😊		...
User 2		😞	😊			...
User 3		😊				...
User 4			😞	😊	😞	...
...	...	...	...	...	...	...

**Tableau 1** : Exemple de matrice utilisateur [26]

Le tableau ci-dessus présente un exemple fictif de matrice binaire contenant des informations de type " l'utilisateur u a apprécié/n'a pas apprécié l'item i ". Ces informations peuvent également être " a acheté/n'a pas acheté ", "a consulté/n'a pas consulté ", etc. Elles peuvent également se mesurer sur un nombre plus élevé de classes : " a mis 1/2/3/4/5 étoiles " etc. Un autre point important est comment le temps influence le profil de l'utilisateur. Les intérêts des utilisateurs, évoluent généralement au cours du temps. Les données du modèle utilisateurs devraient donc constamment être réajustées pour rester conformes aux nouveaux centres d'intérêts de l'utilisateur [26] [6].

### 1.5.3. Extraction de la liste de recommandations

Pour extraire une liste de suggestions c.-à-d. de recommandation à partir d'une matrice utilisateur, les algorithmes utilisent la notion de mesure de similarité entre objets ou personnes décrits par le modèle utilisateur. La similarité a pour but de donner une valeur ou un nombre (au sens mathématique du terme) à la ressemblance entre 2 choses. Plus la ressemblance est forte, plus la valeur de la similarité sera grande. A l'inverse, plus la ressemblance est faible, et plus la valeur de la similarité sera petite [26][6].

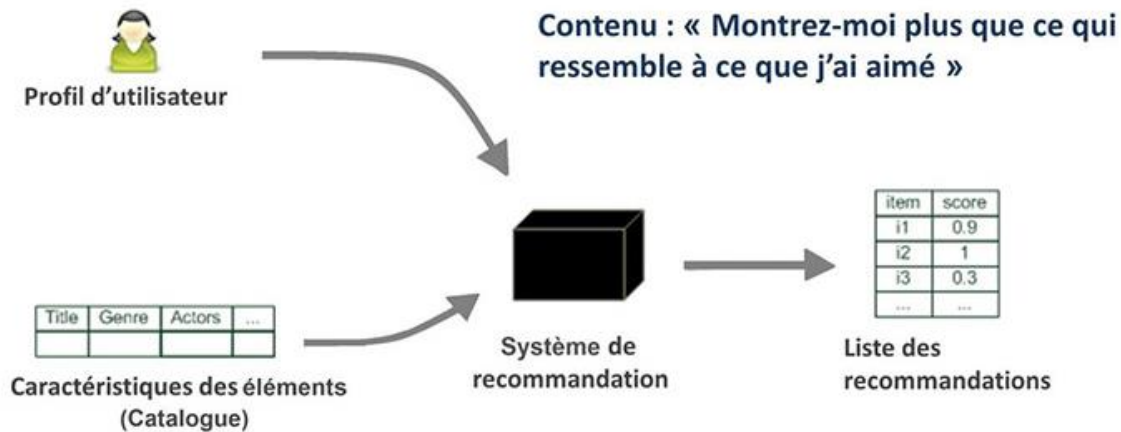
## 1.6. Les principales approches de filtrage

Un système de recommandation doit pouvoir faire des prédictions sur les intérêts des utilisateurs. Il faut donc pouvoir collecter un certain nombre de données sur ceux-ci afin d'être capable de construire un profil pour chaque utilisateur. Nous pouvons distinguer trois grandes classes d'approches selon la nature de ces données.

### 1.6.1. Filtrage basé sur le contenu (content based filtering)

Pour les recommandations basées sur le contenu [31], la tâche consiste à déterminer quels éléments du catalogue coïncident le mieux avec les préférences de l'utilisateur. Une telle

approche ne requiert pas une grande communauté d'utilisateurs ou un gros historique d'utilisation du système. La figure 2 illustre ce processus.



**Figure 2 :** Un système de recommandation basé sur le contenu [24]

La manière la plus simple de décrire un catalogue d'éléments est d'avoir une liste explicite des caractéristiques de chaque élément (on parle aussi d'attributs, de profil d'élément, etc.). Pour un livre par exemple, on peut utiliser le genre, le nom des auteurs, l'éditeur ou toute autre information relative au livre, puis stocker ces caractéristiques (dans une base de données par exemple).

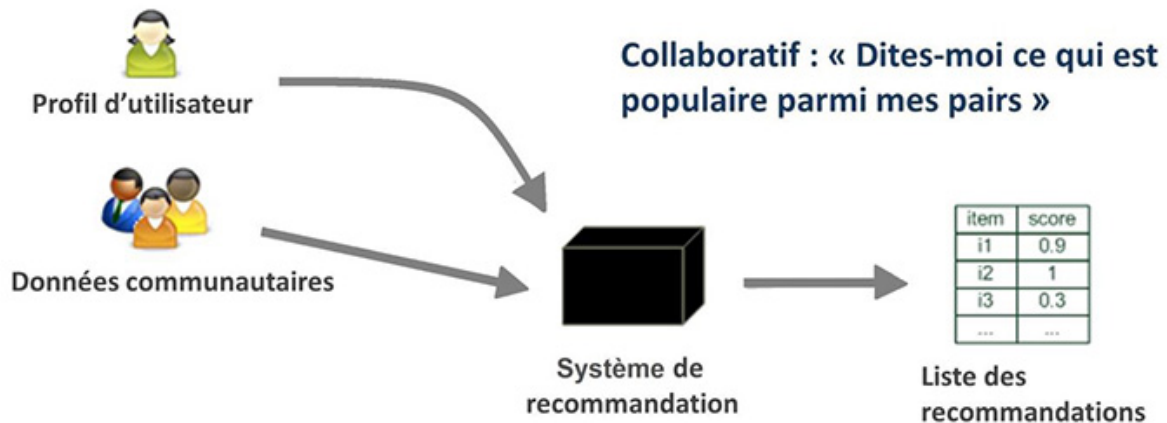
Le profil de l'utilisateur est exprimé sous forme d'une liste d'intérêts basée sur les mêmes caractéristiques. La coïncidence entre les caractéristiques des éléments et le profil de l'utilisateur peut être mesurée de différentes manières :

- l'indice de Dice ou d'autres mesures de similarité ;
- le TF-IDF (*Term Frequency-Inverse Document Frequency*) ;
- les techniques basées sur la similarité des espaces vectoriels (les approches bayésiennes [32], les arbres de décision, etc.) couplées avec des techniques statistiques, lorsqu'il y a trop de mots-clés.

### 1.6.2. Filtrage collaboratif (collaborative filtering)

Les systèmes basés sur le filtrage collaboratif [7] produisent des recommandations en calculant la similarité entre les préférences d'un utilisateur et celles d'autres utilisateurs. De tels systèmes ne tentent pas d'analyser ou de comprendre le contenu des éléments à

recommander. La méthode consiste à faire des prévisions automatiques sur les intérêts d'un utilisateur en collectant des avis de nombreux utilisateurs. L'hypothèse sous-jacente de cette approche est que ceux qui ont aimé un élément spécifique dans le passé auront tendance à aimer cet élément spécifique, ou un autre très « proche », à nouveau dans l'avenir. La figure 3 illustre ce processus



**Figure 3 : Un système de recommandation collaboratif**  
[24]

L'idée des approches collaboratives est d'essayer de prédire l'opinion d'un utilisateur sur les différents éléments. La recommandation est basée sur les goûts et avis précédents de l'utilisateur et sur une mesure de similarité avec d'autres utilisateurs. Les principales étapes de cette approche sont :

1. de nombreuses préférences d'utilisateurs sont enregistrées ;
2. un sous-groupe d'utilisateurs est repéré dont les préférences sont similaires à celles de l'utilisateur qui cherche la recommandation ;
3. une moyenne des préférences pour ce groupe est calculée ;
4. la fonction de préférence qui en résulte est utilisée pour recommander des éléments à l'utilisateur qui cherche la recommandation.

Il est possible de distinguer trois types d'approches pour définir la similitude ou la similarité :

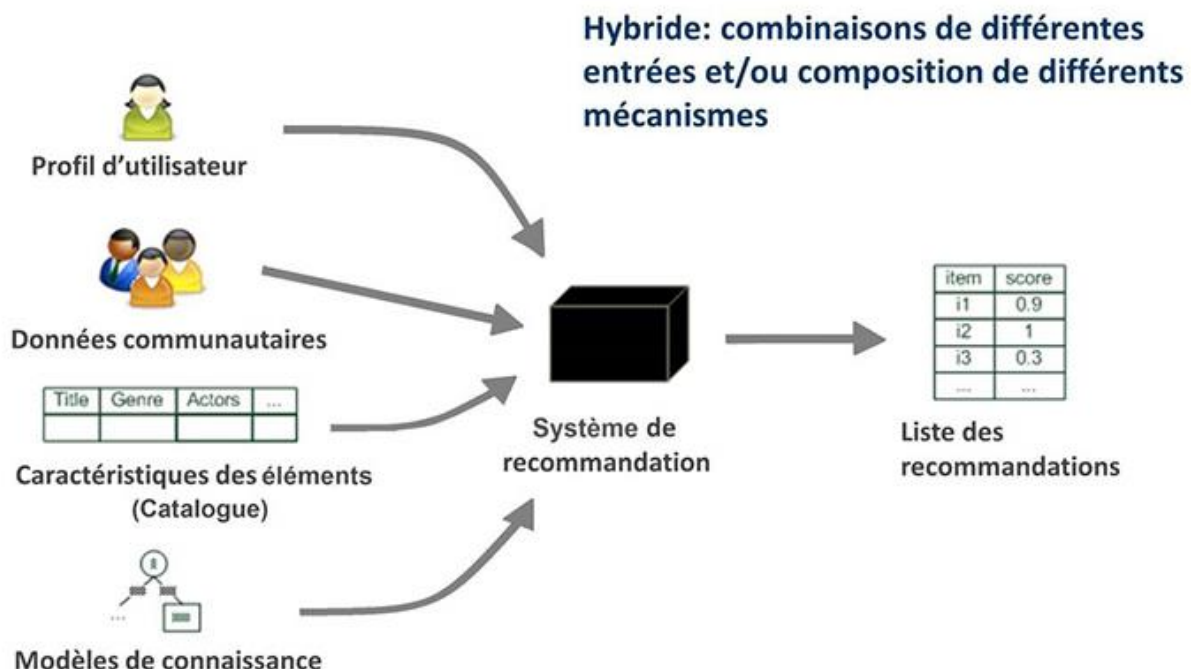
- les approches Item-to-Item basées sur la similarité entre les éléments (*items*). Notons que cette approche s'adapte à un nombre très important d'utilisateurs ou d'éléments.
- les approches User-to-User basées sur la similarité entre les utilisateurs (*users*). Notons que cette approche n'est pas adaptée à un nombre très important d'utilisateurs.
- et les autres approches

Les systèmes de recommandation collaboratifs, par leur diversité, s'appuient donc sur de nombreuses techniques, qu'il s'agisse de :

- similarité entre utilisateurs (coefficient de corrélation de Pearson [35], etc.) ou de sélection de voisinage (les algorithmes basés sur la recherche de voisinage) pour les approches *User-to-User* ;
- similarité entre éléments (la mesure de similarité cosinus [Qam10], etc.) pour les approches *Item-to-Item* ;
- techniques de prédiction de scores (analyse en composantes principales ou ACP, factorisation de matrices, analyse sémantique latente, règles d'association, approches bayésiennes, etc.) pour les autres approches.

### 1.6.3. Filtrage hybride (hybrid filtering)

Un système de recommandation hybride utilise des composants de différents types d'approches de recommandation ou s'appuie sur leur logique [10]. Par exemple, un tel système peut utiliser à la fois des connaissances extérieures et les caractéristiques des éléments, combinant ainsi des approches collaboratives et basées sur le contenu [24].



**Figure 4 :** Le système de recommandation hybride [24]

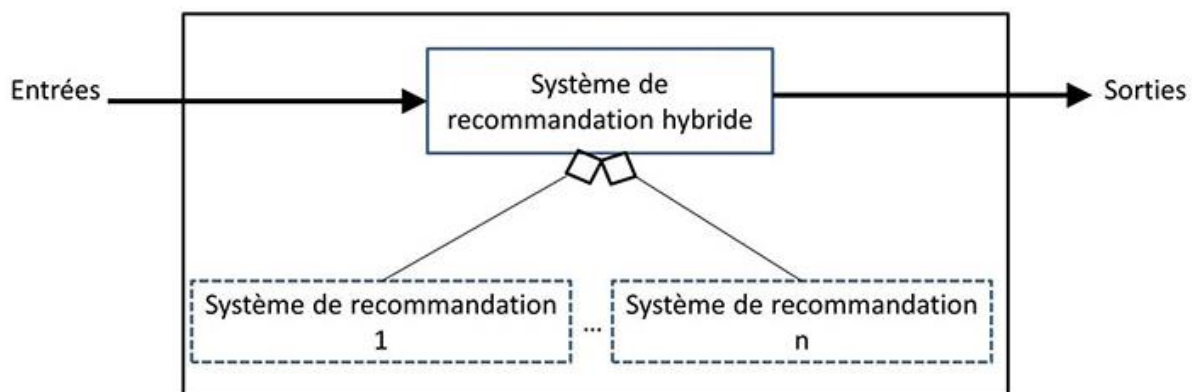
Il est à noter que le terme « hybride » est un artefact de l'évolution historique des systèmes de recommandation où certaines sources de connaissances ont été exploitées en premier lieu,



conduisant à des techniques bien établies qui ont ensuite été combinées. L'objectif est alors de s'appuyer sur des sources de connaissances multiples, en choisissant les plus appropriées à une tâche donnée afin de les utiliser le plus efficacement possible, la figure 4 illustre ce processus.

Il existe trois grandes catégories de combinaisons de systèmes de recommandation pour concevoir un système de recommandation hybride [10] [24] : la combinaison monolithique (*monolithic hybridization design*), la combinaison parallèle (*parallelized hybridization design*) et la combinaison tubulaire (*pipelined hybridization design*).

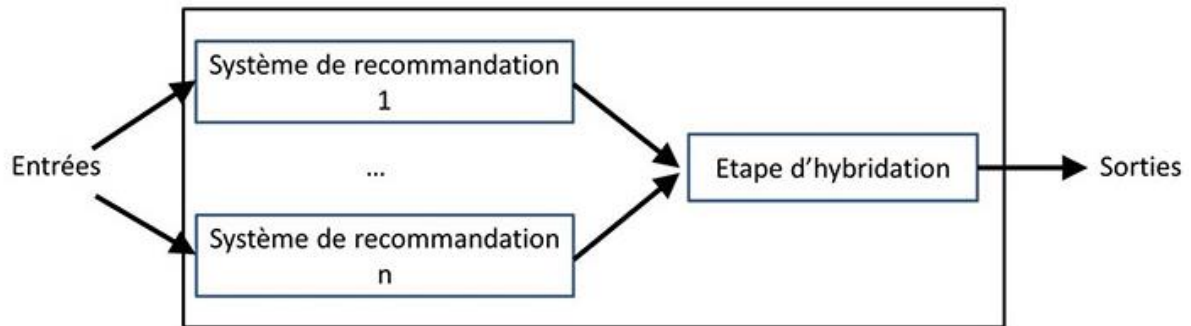
« Monolithique » décrit une conception d'hybridation qui intègre les aspects de différentes stratégies de recommandation en un seul algorithme. Comme illustré sur la figure 5 ci-dessous, différents systèmes de recommandation y contribuent puisque l'approche hybride utilise des données d'entrée additionnelles qui sont spécifiques à un autre algorithme de recommandation, ou bien les données d'entrée sont complétées par une technique et exploitées par une autre. Par exemple, un système de recommandation basé sur le contenu qui exploite également des données communautaires pour déterminer des similarités entre éléments relève de cette catégorie [10] [24].



**Figure 5 :** Conception d'hybridation monolithique

[24]

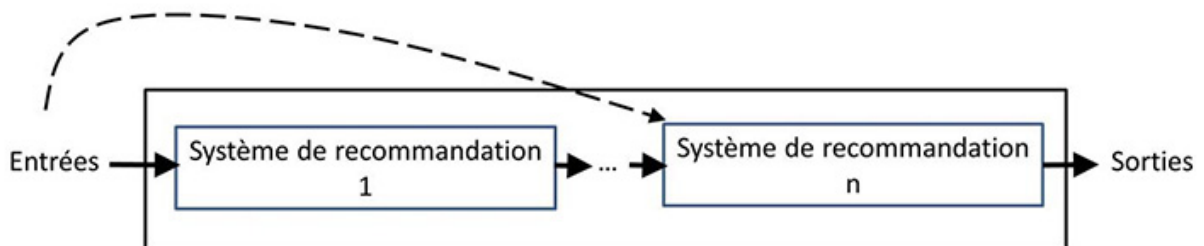
Les deux autres approches hybrides nécessitent au moins deux mises en œuvre de recommandations séparées qui sont combinées en conséquence. Sur la base de leurs données d'entrée, les systèmes hybrides de recommandation parallèles fonctionnent indépendamment l'un de l'autre et produisent des listes de recommandations distinctes, comme illustré sur la figure 6. Dans une étape ultérieure d'hybridation, leurs sorties sont combinées en un ensemble final de recommandations [10] [24].



**Figure 6 :** Conception d'hybridation parallèle

[24]

Lorsque plusieurs systèmes de recommandation sont joints dans une architecture tubulaire, comme illustré par la figure 7, la sortie de l'un des systèmes de recommandation devient une partie des données d'entrée du système suivant [10] [24].



**Figure 7 :** Conception d'hybridation tubulaire

[24]

## 1.7. Avantages et inconvénients des systèmes de recommandations

Malgré leur popularité croissante, les systèmes de recommandation ont subi quelques ratés.

Nous identifions quelques problèmes dont ils souffrent [26] :

- **Adaptabilité :** Au fur et à mesure que la base de données des évaluations augmente, la recommandation devient plus précise.
- **Nouvel utilisateur :** Un nouvel utilisateur qui n'a pas encore accumulé suffisamment d'évaluations ne peut pas avoir de recommandations pertinentes.

- **Nouvel item** : Un item doit avoir suffisamment d'évaluations pour qu'il soit pris en considération dans le processus de recommandation.
- **Démarrage à froid** : Souvent, on se retrouve confronté au problème qu'un utilisateur ne soit comparable avec aucun autre. Ce problème est dû au fait que peu ou pas d'utilisateurs ont évalué un article donné, ou qu'un utilisateur donné a évalué très peu ou pas d'articles. Généralement, ce problème survient quand un nouvel utilisateur ou une nouvelle ressource est ajouté à la base de recommandation.
- **Le cas du système débutant** : provient lors du lancement d'un nouveau service de recommandation. Le système ne possède alors aucune information sur les utilisateurs et sur les items. Les méthodes de filtrage collaboratif ne peuvent pas fonctionner sur une matrice d'usages vide. La solution consiste en général à trouver des informations descriptives des items afin d'organiser le catalogue et inciter les utilisateurs à le parcourir jusqu'à ce que la matrice d'usages soit assez remplie et permette de passer en mode collaboratif. Le tableau 2. résume les forces et faiblesses des méthodes traditionnelles utilisées par les systèmes de recommandation, en l'occurrence le Filtrage Collaboratif (FC), le Filtrage à Base de Contenu (FBC).

Le tableau ci-dessous résume les avantages et inconvénients des méthodes traditionnelles utilisées par les systèmes de recommandation.

Approches	Avantages	Inconvénients
<b>Filtrage à base de</b>	<ul style="list-style-type: none"> <li>- Pas besoin d'une large communauté d'utilisateurs pour effectuer des recommandations.</li> <li>- Une liste de recommandation peut être générée même s'il n'y qu'un seul utilisateur.</li> <li>- La qualité croit avec le temps.</li> </ul>	<ul style="list-style-type: none"> <li>- L'analyse du contenu est nécessaire pour faire une recommandation.</li> <li>- Problème de recommandation des images et de vidéos en absence de Métadonnées.</li> <li>- Nécessité du profil</li> </ul>

<b>contenu</b>	<ul style="list-style-type: none"> <li>- Pas besoin d'information sur les autres utilisateurs.</li> <li>- Prendre en considération les goûts uniques des utilisateurs</li> </ul>	d'utilisateur.
<b>Filtrage collaboratif</b>	<ul style="list-style-type: none"> <li>- Ne demande aucune connaissance sur le contenu de l'item ni sa sémantique.</li> <li>- La qualité de la recommandation peut être évaluée.</li> <li>- Plus le nombre d'utilisateurs est grand plus la recommandation est meilleur</li> </ul>	<ul style="list-style-type: none"> <li>- Démarrage à froid.</li> <li>- Nouvel Item.</li> <li>- Nouvel utilisateur.</li> <li>- Problème de confidentialité.</li> <li>- La complexité dans le système avec un grand nombre d'items et d'utilisateurs le calcul croît linéairement.</li> </ul>

**Tableau 2** : Les avantages et les inconvénients des techniques de recommandation

## 1.8. Evaluation des Systèmes de recommandation

Évaluer un système de recommandation permet de mesurer ses performances vis-à-vis de ses objectifs. De ce fait, les dimensions à évaluer diffèrent selon les objectifs fixés (prédiction des notes des utilisateurs pour les items, augmentation des ventes, etc.)

Il existe une classification des méthodes d'évaluation des systèmes de recommandation. Comme expliqué dans [18], ces méthodes d'évaluation sont classées en trois types : expérimentations offline, études avec des utilisateurs (user studies) et tests réels (real life testing). Ce dernier type est nommé expérimentations en ligne (Online experiments) par [40].

### 1.8.1. Évaluation online

L'évaluation online peut aussi recueillir le point de vue de l'utilisateur concernant le système de recommandation. Dans ce type d'évaluation, des utilisateurs réels utilisent le système dans des conditions réelles sur une longue période [30].

Selon Herlocker [23] ce type d'évaluation peut montrer les usages et les habitudes d'utilisation des utilisateurs, les problèmes et les besoins non satisfaits, et les problèmes que les chercheurs n'ont peut-être pas envisagés dans une étude utilisateurs. Avec ces tests réels sur le terrain, la plupart des objectifs centrés sur l'utilisateur peuvent être efficacement évalués, comme l'évaluation de l'expérience utilisateur, la satisfaction des utilisateurs ou la rétention des utilisateurs [23] [40].

## 1.8.2. Evaluation offline

Une grande partie du travail d'évaluation des algorithmes des systèmes de recommandation s'est concentrée sur l'analyse hors ligne de la précision des prédictions que peuvent faire ces systèmes [23]

Les évaluations hors lignes utilisent des ensembles de données (dataset) constitués d'actions des utilisateurs. Les évaluations offline simulent le processus de recommandation où un sous ensemble des actions utilisateurs du dataset est caché et le système de recommandation prédit ces actions cachées. Le système de recommandation est évalué en fonction de sa capacité à prédire ces interactions cachées. Les résultats de ces prédictions sont analysés en utilisant une ou plusieurs métriques. [44]

### 1.8.2.1. Protocoles d'évaluation

Il en existe trois selon [19] :

- **Retrait (Hold out)** : un pourcentage du dataset est utilisé comme données d'apprentissage tandis que le reste est utilisé comme données de test.
- **Tous sauf 1 (All but 1)** : Un seul item du dataset est utilisé comme données de test, ce qui maximise la quantité de données utilisées comme données d'apprentissage.
- **Validation croisée K-fold (K-fold cross validation)** : Les données d'évaluation sont divisées K fois en données d'apprentissage et en données de test.

### 1.8.2.2. Métriques d'évaluations

Les métriques d'évaluation permettent de mesurer la capacité de l'algorithme de recommandation .ses dernières utilisent des évaluations correctes d'un dataset de test et les comparent avec les résultats prédictions proposées par l'algorithme de recommandation en se basant sur un dataset d'apprentissage.

La majorité de ces métriques proviennent du domaine de la recherche d'information. La précision et le rappel [44] font partie de ces métriques.

- **La précision** : est une mesure d'exactitude. Elle détermine la proportion des items pertinents recommandés parmi tous les items recommandés : autrement dit, la proportion des recommandations qui se sont avérées pertinentes.
- **Le rappel** : est une mesure d'exhaustivité qui détermine la proportion des items pertinents recommandés parmi tous les items pertinents.

D'autres métriques sont utilisées pour évaluer les systèmes de recommandation telle l'erreur quadratique absolue (MAE) et l'erreur quadratique moyenne (RMSE).

- **MAE** : calcule l'écart entre les notes prédites et les vraies notes.
- **RMSE** : elle repose sur le même principe que la MAE, tout en mettant l'accent sur les écarts importants. [19].

## 1.9. Conclusion

Ce chapitre avait pour objectif de faire un tour d'horizon, non exhaustif, sur les systèmes de recommandation. Nous avons évoqué leur historique, une définition théorique de ce qu'un système de recommandation, Nous avons présenté les trois principaux types de systèmes de recommandation : ceux basés sur le contenu, ceux basés sur un filtrage collaboratif, et finalement les approches hybrides. Ensuite, nous avons présenté les avantages et les inconvénients de chaque type de système de recommandation. Aussi la manière d'évaluer ces systèmes.

Parmi les inconvénients majeur est le fait que ces systèmes restent limités car ils ne sont pas capables de s'adapter à leur environnement, autrement dit, ils ne prennent pas en compte le contexte (c'est-à-dire : l'ensemble des éléments qui peuvent influencer la compréhension d'une situation particulière).

À cet effet, les systèmes de recommandation sensible au contexte (les CARS : Context Aware Recommender Systems) sont nés et leur qualité est étroitement liée à leur capacité à prendre en compte le contexte dans lequel se trouve l'utilisateur lorsqu'il désire une recommandation. Nous allons le détailler dans le chapitre 2.

# Chapitre 2

*Les systèmes de  
recommandation  
sensible au contexte*



## 2.1. Introduction

L'objectif des systèmes de recommandation est essentiellement d'améliorer les interactions entre les utilisateurs et les systèmes de recherche et d'accès à l'information. Seulement, les systèmes de recommandation discutés précédemment opèrent dans un espace bidimensionnel (utilisateur, contenu). Or, dans les environnements mobiles, d'autres aspects sont à considérer dans la recommandation. Le profil de l'utilisateur est insuffisant pour produire des recommandations. Il y a plusieurs items dépendants du contexte qui devraient être considérés (quand ?, où ?, pourquoi ?, etc.). En effet, le contexte de l'utilisateur influence le choix de l'information pertinente.

Une nouvelle famille de systèmes de recommandation appelés systèmes de recommandation sensibles au contexte se développe de plus en plus dans la littérature ces dernières années, qui se basent sur le contexte de l'utilisateur en modélisant des données comme sa localisation, l'heure et les personnes aux alentours. Le filtrage consiste ensuite à analyser le contexte courant de l'utilisateur et à sélectionner, parmi les contenus disponibles, ceux qui correspondent le mieux à la situation décrite.

Dans la première partie, nous présenterons les différentes visions de la définition du contexte donnée par quelques chercheurs. Puis nous présentons les concepts de base à ces systèmes, ainsi les différents paradigmes pour son intégration dans les systèmes de recommandation.

La deuxième partie du chapitre sera consacré aux systèmes de recommandation sensible au contexte. Nous donnerons quelques définitions de ces derniers, puis nous exposerons leur architecture. Nous terminerons par quelques exemples sur des SRs sensibles au contexte.

## 2.2 Définition du contexte

Le contexte est une notion vaste pour laquelle il est particulièrement difficile de donner une définition générale et opérationnelle.

En informatique, plusieurs définitions ont vu le jour par plusieurs chercheurs. Dans notre travail, nous avons sélectionné quelques-unes qui ont été marquées au fil des années.

En premier, les chercheurs ont débuté par définir le contexte en énumérant tous les éléments (ces éléments constituent des informations non fonctionnelles de l'application, mais qui peuvent apporter un impact sur cette dernière) qui peuvent être considérés comme une information du contexte. Parmi ces définitions nous trouvons :

Schilit et Theimer [40] qui définissent le contexte comme étant «*la localisation et l'identité des personnes et objets à proximité ainsi que les modifications pouvant intervenir sur ces objets*».

Dans le domaine des systèmes de recommandation, la définition retenue est celle de Anind K. Dey et Gregory D. Abowd [16], Leur étude vise à donner la définition la plus précise possible. Pour eux, «*le contexte est l'ensemble de toutes les informations qui peuvent être utilisées pour caractériser la situation d'une entité* ». Pour résumer, une information fait partie du contexte si elle influe sur une interaction entre deux entités, une entité pouvant être un acteur, un lieu, un objet de l'environnement considéré comme utile à l'interaction entre la personne et le système.

### 2.3 Sources d'informations contextuelles

On dénote trois grands types de sources d'informations contextuelles : explicite, implicite ou inférée [42].

- **Explicite** : l'information sur le contexte est déjà incluse dans les données ou directement demandée à l'utilisateur. Par exemple, sur des plateformes d'achats en ligne, on peut demander à l'utilisateur s'il effectue un achat pour des raisons personnelles ou professionnelles ou encore pour offrir un cadeau au moment de l'achat. On peut aussi, lors de l'inscription de l'utilisateur sur le système, lui demander de remplir un formulaire contenant des informations personnelles le concernant, telles que son âge, sa profession, etc.
- **Implicite** : l'information est obtenue à partir des données ou de l'environnement dans lequel se trouve un utilisateur sans la lui demander explicitement. Pour des plateformes de recommandation de sites touristiques à partir des Smartphones, on peut par exemple connaître la situation géographique exacte d'un individu au moment d'effectuer une recommandation.
- **Inférée** : l'information est obtenue à l'aide de méthodes d'exploitation et d'exploration des données. Par exemple, l'identité d'une personne parcourant les chaînes de télévision peut ne pas être explicitement connue pour une société de télévision par câble. Cependant, le système peut arriver à apprendre le moment de la journée, la chaîne et le type de programme regardés par les différents utilisateurs d'un même foyer (père, mère, enfants...) et ceci avec une précision acceptable en utilisant des techniques de data mining.

## 2.4 Caractéristiques du Contexte

Dans ce qui suit, nous allons donner des caractéristiques techniques de l'information du contexte :

- **La Continuité** : Pendant que l'utilisateur se déplace d'une localisation à une autre ce qui implique un changement dans la valeur du contexte, ou bien quand l'utilisateur obtient de nouvelles ressources, le contexte évolue. Cette évolution doit s'accompagner d'un fonctionnement continu des systèmes malgré les changements incessants du contexte.
- **L'hétérogénéité** : Une large hétérogénéité en termes de modélisation, de traitement et de qualité provient du fait que le contexte est capturé à partir d'une variété de sources [22]. Par exemple, en informatique, les ordinateurs portables, les Smartphones, les tablettes sont différentes à tous niveaux (Matériels, logiciels, communications...etc.).
- **L'imprévisibilité** : L'une des questions qui se pose est de savoir comment un système doit prendre en compte l'imprévisibilité des changements de l'environnement et les interactions de l'utilisateur. Car certains éléments du contexte peuvent ne pas être connus à l'avance dans le système.
- **L'interdépendance** : L'information et/ou changement de valeur de l'information de contexte, peut être dépendante d'une autre information et/ou changement de valeur de l'information de contexte [7].
- **Contexte imparfait** : cette caractéristique est déterminée selon la source de l'information. Le contexte peut donc être imprécis ou bien le contexte peut être erroné ou même inconnu.

## 2.5 Dimension du Contexte

Selon le travail effectué par M. Daoud, on distingue deux dimensions principales du contexte de recherche en premier lieu le contexte de l'utilisateur et en second lieu le contexte de la requête. Elle vise à intégrer ces deux dimensions du contexte dans un processus de recherche d'information (ou recommandation) dans le but d'affermir un système de recherche d'information contextuel. La première permet de caractériser l'utilisateur à travers ses centres d'intérêts à court et à long terme et ses préférences. La seconde dimension est liée à la requête et agréée de déterminer la nature des résultats attendus par l'utilisateur.

## 2.6 Systèmes sensibles au contexte

Le terme sensibilité au contexte est devenu aujourd'hui une technologie embarquée dans une grande variété de systèmes informatiques, et nous pouvons en citer diverses définitions comme suite :

La notion de sensibilité au contexte a été tout d'abord introduite dans le domaine de l'IHM (Interactions Homme-Machine) par Weiser en 1991 pour réconcilier le monde virtuel et le monde physique [45]. Il déclare que les technologies les plus réussies sont celles qui s'associent à la vie de tous les jours, jusqu'à ce qu'il devienne difficile de distinguer les deux. L'informatique ubiquitaire consiste en l'intégration du monde informatique partout dans l'environnement et de manière invisible.

La première définition de la sensibilité au contexte a été proposée par [38], comme étant la capacité d'un système à s'adapter au contexte de son exécution en fonction de sa localisation, de l'ensemble des personnes à proximité, des équipements accessibles, etc. [9] annonce qu'une application sensible au contexte doit automatiquement extraire de l'information ou effectuer des actions en fonction du contexte utilisateur détecté par les capteurs. [35] définit la sensibilité au contexte comme étant la capacité des dispositifs informatiques à détecter, interpréter et répondre aux aspects de l'environnement local d'un utilisateur et des dispositifs informatiques eux-mêmes. [15] affine ces définitions en une autre plus générale. Il annonce qu'un système est sensible au contexte lorsqu'il utilise le contexte pour fournir des informations pertinentes aux utilisateurs : *"a system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task"*.

Ces définitions s'articulent toutes autour de l'aptitude d'un dispositif à adapter son comportement en se basant sur le contexte de l'utilisateur. Un système sensible au contexte est un ensemble de mécanismes destinés pour la collection et la gestion des informations de contexte, et le contrôle du comportement du système en fonction de ces informations. Nous présentons dans ce qui suit l'étude de l'architecture dans le domaine de la sensibilité au contexte.

## 2.7. Approches d'intégration du contexte dans les systèmes de recommandation

La contextualisation peut intervenir à différents endroits dans un système de recommandation. Selon U. Panniello et al. [34], il existe trois différents paradigmes algorithmiques pour l'intégration des informations contextuelles dans le processus de recommandation sont présentés :

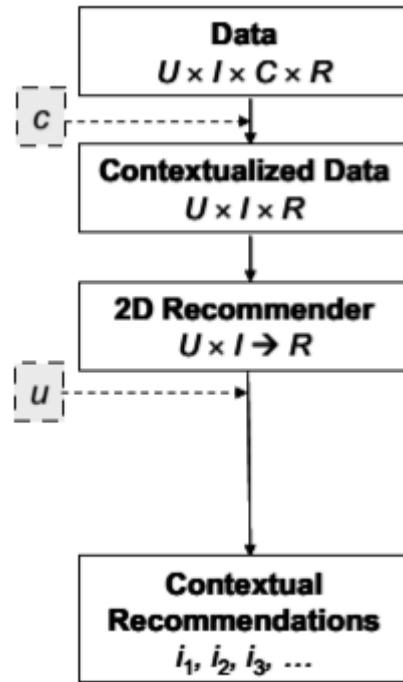
1. L'approche de Pré-filtrage Contextuel (*contextual pre-filtering*),
2. L'approche de Post-filtrage Contextuel (*contextual post-filtering*),
3. L'approche de la Modélisation Contextuelle (*contextual modeling*).

### 2.7.1 Approche de Pré-filtrage Contextuel (contextual pre-filtering)

Dans l'approche de pré-filtrage, l'information contextuelle est utilisée comme une étiquette permettant de filtrer les évaluations qui ne correspondent pas à l'information contextuelle spécifiée. Cela se fait avant que la méthode principale de recommandation soit lancée sur le reste des données sélectionnées.

Autrement dit, si un contexte d'intérêt particulier est  $k$ , alors cette méthode sélectionne à partir de la série initiale toutes les évaluations relatives au contexte spécifié  $k$ , et elle génère la matrice «Item x utilisateur » ne contenant que les données relatives au contexte  $k$ .

Après cela, la méthode des systèmes de recommandation (SR), comme le filtrage collaboratif, est lancée sur la base de données réduite afin d'obtenir les recommandations liées au contexte  $k$ . La figure 8 montre l'intégration du contexte dans un système de recommandation à deux dimensions (utilisateur x produit).



**Figure 8 :** Approche Pré-filtrage Contextuel [5]

Il existe différents types de pré-filtrage, tels que le pré-filtrage exact (Exact Pre-Filtering) et le pré-filtrage généralisé (Generalized Pre-Filtering). Exact Pre-Filtering (EPF) sélectionne toutes les opérations relatives exactement au contexte spécifique, tandis que Generalized Pre-Filtering sélectionne toutes les opérations visées à un contexte spécifique basée sur la généralisation de l'information contextuelle (par exemple, un "Cadeau pour ses parents" peut être généralisé à "cadeau pour ses relations").

### 2.7.2. Approche de Post-filtrage Contextuel (contextual post-filtering)

Selon l'approche post-filtrage, l'information contextuelle est utilisée après le lancement de la méthode principale de recommandation 2D (à deux dimensions). Une fois les évaluations inconnues sont estimées et les recommandations sont produites, le système analyse les données pour un utilisateur donné dans un contexte précis pour trouver les modèles d'utilisation des articles spécifiques, et utilise ces modèles pour «contextualiser» les recommandations obtenues à partir de la méthode classique de recommandation 2D, comme le filtrage collaboratif. La figure au-dessous montre comment utiliser le contexte après la recommandation (2D).

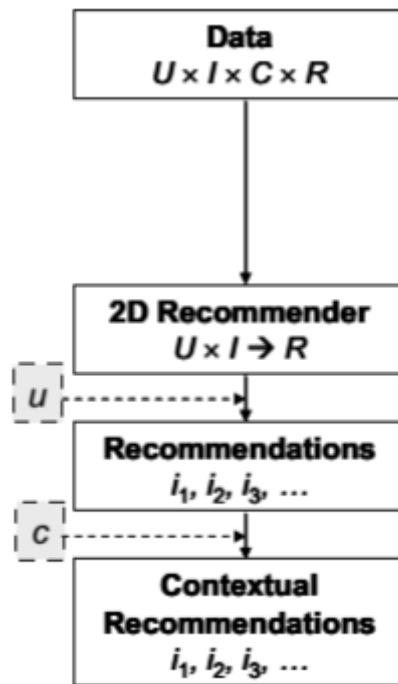


Figure 9 : Approche Post-filtrage contextuel [5]

### 2.7.3. Approche de la Modélisation Contextuelle (contextual modeling)

En plus des deux méthodes citées auparavant, il existe une troisième méthode à savoir la modélisation contextuelle dans laquelle l'information contextuelle est utilisée directement à l'intérieur des algorithmes de génération des recommandations. La figure 10 montre l'intégration du contexte dans le processus de recommandation.

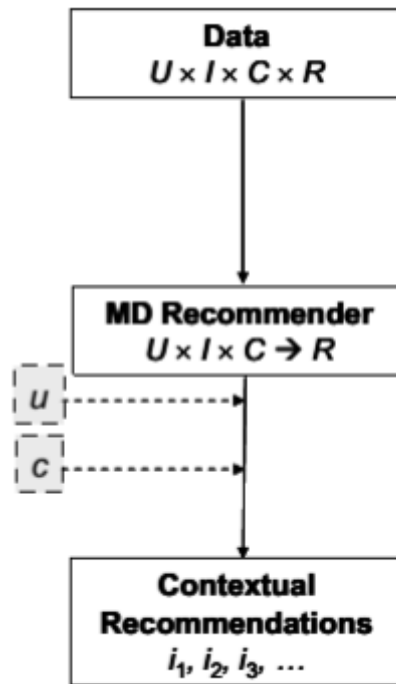


Figure 10 : Approche Modélisation contextuelle [5]

## 2.8. Exemples de Système de Recommandation sensibles au contexte

Plusieurs travaux ont été effectués sur les systèmes de recommandation introduisant la notion du contexte. Nous présentons, dans cette section, quelques systèmes de recommandation sensibles au contexte.

### 2.8.1. Micro-profiling

L'approche micro-profiling [3] a pour but de recommander des chansons/artistes inconnus à un utilisateur. Le type du système de recommandation est le **pre-filtering** car les données sont filtrées selon le contexte (le temps : moment de la journée, jour, mois ou année).

Dans cette approche, supposant que les goûts des utilisateurs changent selon une période et peuvent être similaires dans une même période.

Par exemple, un utilisateur préfère écouter un certain genre de chanson quand il travaille, et un autre genre de chanson avant de dormir.

On dispose d'un profil utilisateur dans lequel on affectera les chansons/artistes écoutés par cet utilisateur. Ce profil sera décomposé de façon à ce qu'il s'adapte à chaque contexte : dans cette approche, le temps représente le moment de la journée, il sera divisé en plusieurs segments en fonction des chansons que l'utilisateur écoute le plus durant chaque moment (segment de temps) de la journée.



L'utilisation de ces micro-profiles améliore l'exactitude de la recommandation. L'un des problèmes posés par cette méthode, réside dans le fait que le partitionnement du temps diffère d'un utilisateur à un autre (au lieu d'être fixe pour tous les utilisateurs), ce qui risque de diminuer la précision du système.

### 2.8.2. Amazon

Un autre système de recommandation sensible au contexte est celui d'**Amazon** [48] qui a été créé par Jeff Bezos en juillet 1995 et dont la filiale française a ouvert en 2000. **Amazon.com** est une entreprise de commerce électronique américaine basée à Seattle. Sa spécialité la plus connue est la vente de livres, mais elle est diversifiée dans d'autres produits, notamment dans la vente de tout type de produits culturels : disques CD, musique en téléchargement, DVD, appareils photos numériques, matériels informatique et électroménager, etc. Ce système demande aux utilisateurs de se connecter via leur nom comme c'est démontré dans partie A de la figure 11 afin de créer leur profile et fournit un bouton « trouver un cadeau » (voir partie B de la figure 11) pour chaque utilisateur afin de faire la distinction entre les préférences propres à l'utilisateur et les préférences de la personne à laquelle il va offrir ce cadeau.

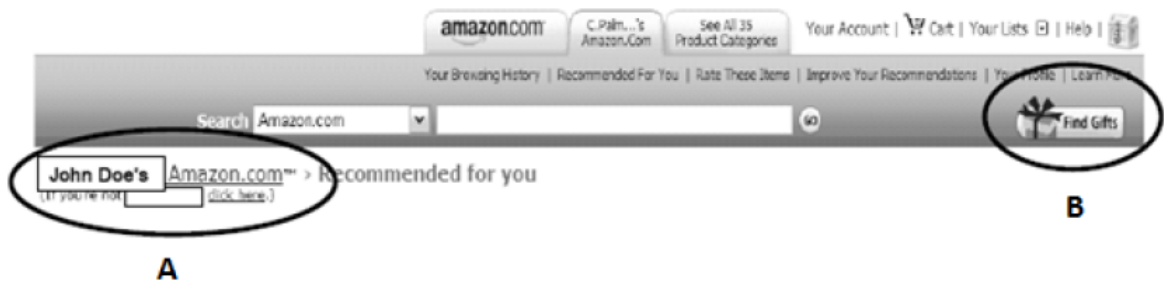


Figure 11 : Barre d'outils du site [www.amazon.com](http://www.amazon.com) [48]

### 2.9. Splitting approche du pré-filtrage contextuel

Parmi les algorithmes proposés dans les CARS, nous nous intéressons aux algorithmes de l'approche de découpage. Cette approche appartient à l'approche de pré-filtrage contextuel des CARS.

Cette approche consiste à ne pas filtrer les données mais plutôt de découper (diviser) l'ensemble des données en deux sous-ensembles par rapport à la valeur d'une (des) dimension (s) contextuelle (s).

Ce découpage (Splitting) ne se fait que si les utilisateurs évaluent différemment un item dans deux conditions contextuelle et ce en utilisant des tests statistiques (critères d'impureté (impurity criteria)). Cette approche sera détaillée dans le prochain chapitre.

## **2.10. Conclusion**

Ce chapitre présente le concept de la sensibilité au contexte, un aspect fondamental dans l'informatique ubiquitaire. Nous avons présenté des définitions du contexte, la sensibilité au contexte présenter éventuellement quelques exemples sur des SRs sensibles au contexte, et aussi parler des différents paradigmes utilisés dans les systèmes de recommandation sensible au contexte.

La sensibilité au contexte est un élément central de plusieurs systèmes. Bien que le domaine de l'informatique contextualisé soit déjà en pleine expansion, de plus en plus de capteurs prolifèrent partout dans le monde, sa croissance va s'accélérer et son importance ne fera que croître.

Le prochain chapitre sera consacré à la partie conception et réalisation de notre système de recommandation sensible au contexte

# Chapitre 3

*Implémentation de  
l'approche de  
découpage*

### 3.1. Introduction

La prise en considération du contexte par les systèmes de recommandation permet une meilleure modélisation du comportement des utilisateurs, en fournissant des informations supplémentaires qui peuvent être exploitées afin d'effectuer de meilleures recommandations.

Parmi les techniques utilisées dans l'approche de pré-filtrage contextuel, nous nous intéressons à l'approche de découpage.

Ce chapitre, est consacré à détailler cette approche ainsi son expérimentation sur un dataset du domaine. Nous finirons ce chapitre, par des comparaisons avec des algorithmes des systèmes de recommandation traditionnels.

### 3.2. Terminologies utilisées

Avant d'expliquer l'approche de découpage, nous détaillons quelques terminologies utilisées afin de mieux comprendre son principe. Pour cela, nous utilisons un extrait d'un dataset pour démontrer chacune des terminologies

						Dimension contextuelle
User	Item	Rating	Time	Location	Compagnion	
U1	I1	5	Weekend	Home	Friend	Condition contextuelle
U2	I1	5	Weekday	Cinema	Kids	
U3	I2	4	Weekend	Cinema	Family	
U1	I3	2	Weekday	Home	Friend	
U2	I2	3	Weekend	Home	Family	
U1	I2	?	Weekday	Cinema	Friend	situation contextuelle

**Tableau 3 :** Exemple de jeu de donnée

- **Dimensions (ou facteurs) contextuelles :**

Les variables qui décrivent une situation contextuelle. Exp : *Time, Location, Companion*

- **Conditions contextuelles**

Les valeurs que peut prendre une dimension contextuelle. Exp : *Weekend/Weekday(Time)*

- **Situations contextuelles**

L'ensemble des conditions contextuelles. Exp : { **Weekday, Cinema, Friend** }, { **Weekend, Home, Family** }

### 3.3. Description de l'approche de découpage

Elle consiste à découper soit les items ou les utilisateurs ou les deux dans la matrice de notation en fonction du contexte.

Cette approche améliore la prédiction des ratings du filtrage collaboratif des systèmes de recommandation classiques en prenant en considération les valeurs des différentes dimensions contextuelles, elle a été introduite par [5]

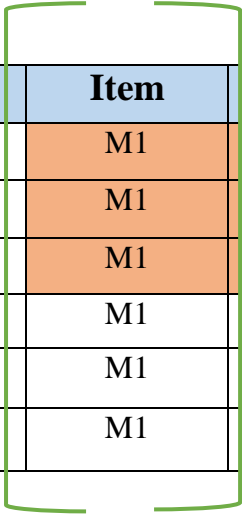
Nous distinguons trois types de Splitting, à savoir: Item Splitting, User Splitting, User and Item Splitting.

#### 3.3.1. Algorithmes de découpage

##### 3.3.1.1. Découpage par item

Proposé par Baltrunas et al., en 2009 Le contexte dépendra de l'item, c-à-d selon les conditions contextuelles, un item est considéré comme deux items différents si la notation change d'une condition à une autre [3].

Exemple :



User	Item	Season	Location	Companion	Rating
U1	M1	Winter	Theater	Family	5
U2	M1	Winter	Theater	Family	5
U3	M1	Winter	DVD House	Friend	4
U1	M1	Summer	Home	Friend	2
U2	M1	Summer	Home	Friend	3
U1	M1	Fall	DVD House	Alone	2

Après avoir calculé la condition contextuelle la plus significatif, nous trouvons que le facteur contextuel season = winter à des notations élevés par rapport aux autres valeurs, donc les items vont être divisé par rapport à cette valeur, dont nous allons obtenir une matrice à 2 dimension.

User	Item	Rating
U1	M11	5
U2	M11	5
U3	M11	4
U1	M12	2
U2	M12	3
U1	M12	3

**Tableau 4 :** Matrice de notation résultat découpage par item

### 3.3.1.2. Découpage par utilisateur

Dans cette approche, le contexte dépendra des utilisateurs càd selon les conditions contextuelles, un user est considéré comme deux utilisateurs différents [38].

Exemple :

User	Item	Season	Location	Companion	Rating
U1	M1	Winter	Theater	Family	5
U2	M1	Winter	Theater	Family	5
U3	M1	Winter	DVD House	Friend	4
U1	M1	Summer	Home	Friend	2
U2	M1	Summer	Home	Friend	3
U1	M1	Fall	DVD House	Alone	2

De la même manière, nous remarquons que la dimension (le facteur) contextuelle à des notations élevés si companion= Family et moins si companion = NonFamily

User	Item	Rating
U11	M1	5
U21	M1	5
U32	M1	4
U12	M1	2
U22	M1	3
U12	M1	3

**Tableau 5 :** Matrice de notation résultat découpage par item

### 3.3.1.3. Découpage par item&utilisateur

Cette approche fusionne les deux approches précédentes [48].

Exemple :

User	Item	Rating
U1	M11	5
U2	M11	5
U3	M11	4
U1	M12	2
U2	M12	3
U1	M12	2

+

User	Item	Rating
U11	M1	5
U21	M1	5
U32	M1	4
U12	M1	2
U22	M1	3
U12	M1	2

=

User	Item	Rating
U11	M11	5
U21	M11	5
U32	M11	4
U12	M12	2
U22	M12	3
U12	M12	2

Tableau 6 : Matrice de notation résultat découpage par item&utilisateur

### 3.3.2. Critères d'impuretés

**tmean, tprop, tG, tch and trandom.** Ces critères s'utilisent dans les arbres de décisions (à nos connaissances).

- **tmean(i, c)** : les critères d'impureté sont définis à l'aide du test-t de deux échantillons et calcule la différence significative.

Plus la valeur t du test est grande, plus la différence de moyens dans les deux partitions est vraisemblable.

$$t_{mean} = \left| \frac{\mu_{i_c} - \mu_{i_{\bar{c}}}}{\sqrt{s_{i_c}/n_{i_c} + s_{i_{\bar{c}}}/n_{i_{\bar{c}}}}} \right|$$

$\mu_i$  : rating moyen de l'item i

$s_i$  : la variance de notation de l'item i

$n_i$  : le nombre de rating que l'item à reçu

- **tprop(i, c)** : connu sous le nom de z-test

$$t_{prop} = \frac{p_{i_c} - p_{i_{\bar{c}}}}{\sqrt{p(1-p)(1/n_{i_c} + 1/n_{i_{\bar{c}}})}}$$

$$\text{Où : } P = (p_{i_c}n_{i_c} + p_{i_{\bar{c}}}n_{i_{\bar{c}}})/(n_{i_c} + n_{i_{\bar{c}}})$$

$p_{i_c}$  : est la proportion de notes élevées

$n_{i_c}$  : est le nombre d'évaluations dans  $i_c$

- **tIG(i, c)** : connu aussi comme la divergence de Kullback-Leibler ,

$$t_{IG} = H(i) - H(i_c)P_{i_c} + H(i_{\bar{c}})P_{i_{\bar{c}}}$$

Où

$H(i)$  : l'Entropie de Shannon

$P_{i_c}$  est la proportion des notes que  $i_c$  reçoit de l'item  $i$ .

- **Tchi(i,c)** : Ce critère est similaire à  $t_{prop}(i, c)$ , mais il utilise une statistique de proportion différente. Il calcule le test du chi carré pour les proportions afin de déterminer s'il existe une différence significative entre les proportions de haute et basse notations
- **trandom(i, c)** : est utilisé comme base de référence pour comparer le comportement des autres méthodes. Il renvoie un score aléatoire pour chaque division  $c$ .

### 3.4. Implémentation de l'approche de découpage

Nous allons présenter dans ce qui suit la démarche détaillée d'obtention d'une recommandation contextualisée utilisant l'approche de découpage. Dans notre travail, nous nous sommes focalisé seulement sur l'algorithme de découpage par item. Les autres seront comme travaux en perspectives. Dans ce qui suit la démarche à suivre et qui correspond à notre architecture :



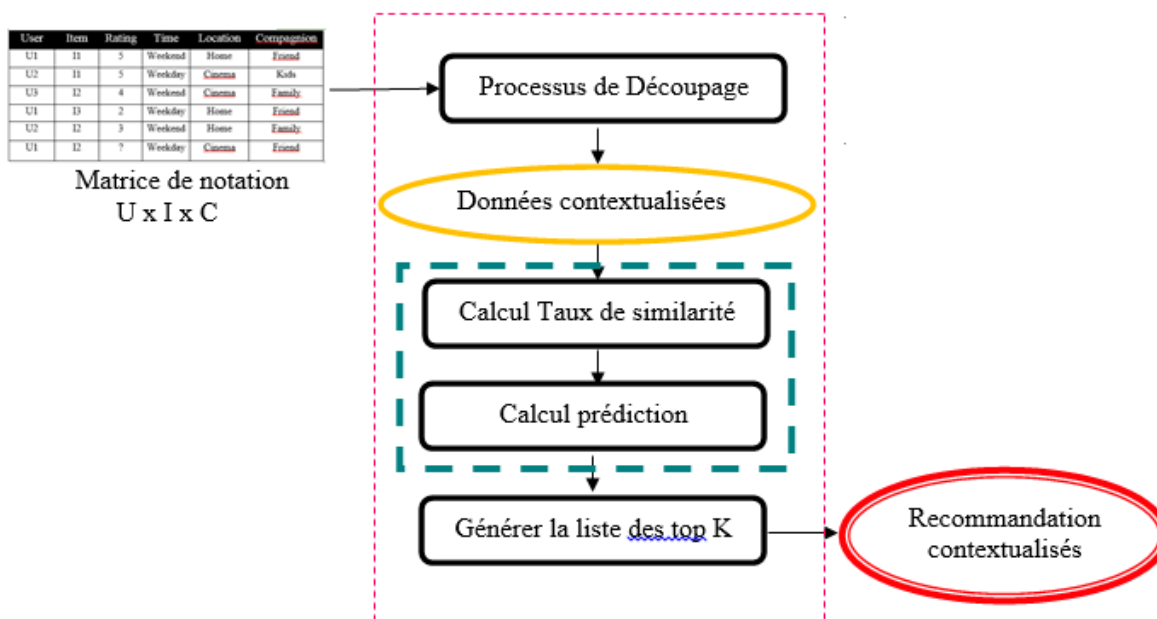


Figure 12 Architecture de l'approche de découpage

### 3.4.1. Processus de découpage

Puisque l'approche appartient au pré-filtrage contextuel, nous allons aboutir à des données contextualisées pour lancer après l'un des algorithmes traditionnels des systèmes de recommandations. Pour cela, comme mentionné, nous avons au départ des données de notation incluant les informations contextuelles.

Dans cette étape, pour obtenir des données contextualisées, nous allons fractionner les items en utilisant une simple division, ce qui implique l'utilisation d'une seule condition contextuelle pour la division car l'utilisation de plusieurs conditions augmente la fragmentation des données ainsi augmente le taux de sparsité.

Pour cela, pour chaque item  $i$ , nous itérons chaque condition contextuelle de chaque dimension contextuelle. Si l'item  $i$  présente des différences significatives (en utilisant des critères d'impureté) dans la matrice de notation, nous créons deux nouveaux items artificiels, puis nous divisons le vecteur de sa notation en deux vecteurs. Dans la phase de test, les prédictions d'évaluation pour l'élément fractionné sont calculées pour l'un des éléments nouvellement générés.

Pour les premiers résultats, nous utilisons  $t_{\text{mean}}$  comme critère d'impureté. Nous n'obtenons en sortie de cette étape que des données contextualisées et une matrice de notation bidimensionnelle.

Dans les systèmes de recommandation négligeant le contexte dis traditionnel seulement l'étape suivante sera faite pour calculer la prédiction.

### 3.4.2. Calcul taux de similarité

Comme algorithme de recommandation traditionnel, nous allons utiliser filtrage collaboratif à base d'utilisateur. De ce fait, nous allons calculer le taux de similarité qui existe entre les différents utilisateurs pour trouver les k plus proche voisin (KNN (K nearest Neighborhood)). Nous allons utiliser la corrélation de Pearson (PCC) dont sa formule est la suivante :

$$Sim_{u,v} = \frac{\sum_{i \in I_{u,v}} (\ddot{r}_{u,i} - \bar{r}_u) \cdot (\ddot{r}_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I_{u,v}} (\ddot{r}_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{i \in I_{u,v}} (\ddot{r}_{v,i} - \bar{r}_v)^2}}$$

$I_{u,v}$  : L'ensemble commun des items notés par l'user actif u et les autres users

$\bar{r}_u, \bar{r}_v$  : La moyenne des notes de u et v

$r_{u,i}$  : La note donnée par user u à l'item i

### 3.4.3. Calcul de prédiction

Une fois nous avons sélectionner les K similaires utilisateurs, nous pouvons prédire les notes pour les items non affecté (pas encore noté) par l'utilisateur u. L'équation suivante permet de calculer la note prédite pour l'item i à l'utilisateur cible u dans le contexte courant.

$$\check{r}_{u,ic} = \bar{r}_u + \frac{\sum_{v \in V_u} sim_{u,v} (r_{v,ic} - \bar{r}_v)}{\sum_{v \in V_u} sim_{u,v}}$$

$V_u$  : L'ensemble des K similaire users

$\bar{r}_u, \bar{r}_v$  : La moyenne des notes de u et v

### 3.4.4. Top N Recommendation

Dans cette étape, l'algorithme sélectionne la liste des N items en tant que liste de recommandations contextualisées à suggérer à l'utilisateur actif c-à-d u.

## 3.5. Evaluation

### 3.5.1. Description du Dataset

Nous avons utilisé un Dataset qui se compose de l'ensemble des users, Items, ratings et une dimension contextuelle (TypeDay) et deux conditions contextuelles (Weekend, Weekday). Ce dernier contient 5035 votes (ratings), 97 users, 79 Items

### 3.5.2. Méthode d'évaluation

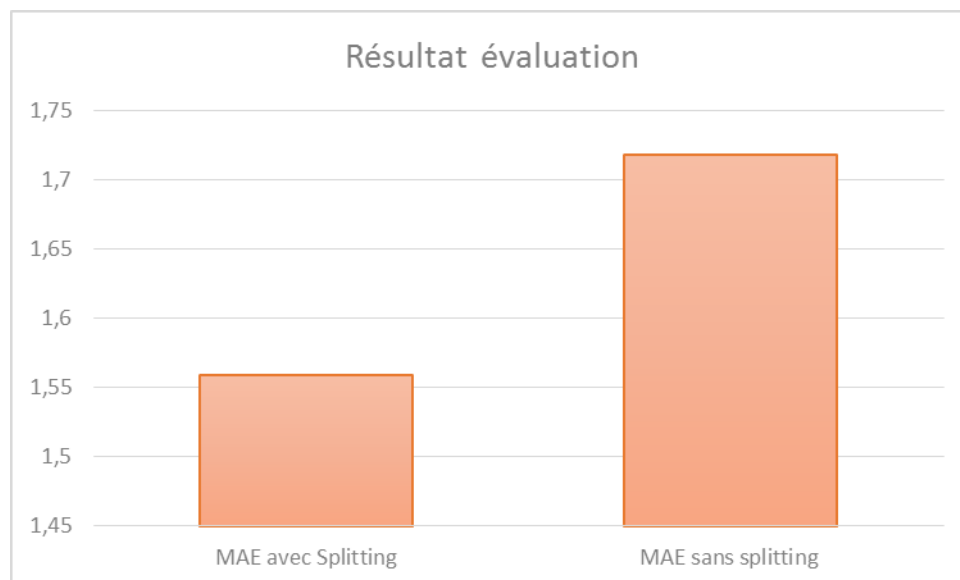
Comme nous avons cité précédemment dans le premier chapitre dans la partie évaluation des systèmes de recommandations, il existe différents protocoles d'évaluations dont nous avons choisi parmi eux le protocole hold out qui divise le dataset en deux parties (training, testset) selon un pourcentage donné, nous avons fixé ce pourcentage à 70% c-à-d, il va mettre les 70% du data set dans la partie training et les 30% restant dans la partie testset

Et parmi les différentes métriques citées dans le premier chapitre, nous avons utilisé MAE qui calcule l'écart entre la valeur réelle et la valeur prédite.

### 3.5.3. Résultat

Après avoir lancé notre application nous avons obtenu les résultats suivants :

9 % des items du fichier d'apprentissage ont été tranchés, ce qui confirme que la notation des items est influencée par le contexte.



**Figure 13 :** Résultat d'évaluation

D'après la figure13, Nous avons remarqué que les résultats d'évaluations obtenus avec Item Splitting s'approche de la valeur 1 ce qui confirme l'utilité de l'approche Item Splitting contrairement à l'autre évaluation sans Item Splitting .

$$MAE = \frac{\sum_{u,i} |r_{u,i} - p_{u,i}|}{N}$$

$r_{u,i}$  : la note donnée par usagé u à l'item i.

$P_{u,i}$ : la note prédite par le système à l'item i .

N :le nombre de notes prédites par le système.

### 3.6. Conclusion

Dans ce chapitre, nous avons détaillé le processus de découpage ainsi que les différentes formules utilisées pour le calcul de prédiction, enfin nous avons décrit notre dataset et la méthode utilisée pour l'évaluation de notre système de recommandation.

Dans le prochain chapitre nous allons vous faire part de l'utilité d'une bibliothèque pour l'évaluation des systèmes de recommandation traditionnel que ceux sensibles aux contextes

# Chapitre 4

*Utilisation de la  
Bibliothèque  
CarsKit*

## 4.1. Introduction

CARSKIT est un moteur de recommandation sensible au contexte et basé sur java et open source, où il peut être utilisé, modifié et distribué sous les termes de la licence public générale (Java version 1.7 ou supérieure requise). Il est spécialement conçu pour les recommandations tenant compte du contexte [47].

CARSKIT est un logiciel gratuit vous pouvez le redistribuer et /ou le modifié selon les termes de la licence public générale [2]

## 4.2. Conception

CARSKIT fournit une architecture flexible permettant d'élargir facilement la portée des algorithmes de recommandations tenant compte du contexte, ainsi que des espaces pour développer de nouveaux algorithmes à l'avenir. L'architecture est représentée par la figure15

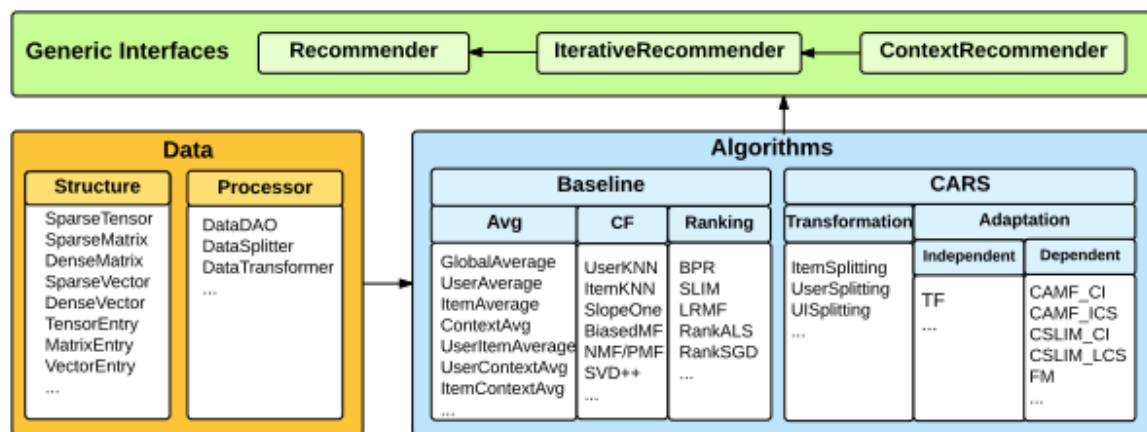


Figure 14 : Architecture de CARSKIT [2]

Le Processus est simple dans notre conception : les différents algorithmes de recommandation sont les implémentations spécifiques et les extensions des interfaces génériques dans lesquelles les fonctions partagées et communes sont définies, telles que la notation ou la prévision du score d'un utilisateur pour un élément dans un contexte spécifique [2]

## 4.3. Algorithmes

Nous divisons les algorithmes contextuels en deux catégories dans CARSKit: algorithmes de transformation et algorithmes d'adaptation

### 4.3.1. Algorithmes de transformations

Les algorithmes de transformation essaient de prétraiter les données et de convertir le jeu de données contextuelles en une matrice de notation bidimensionnelle qui ne contient que des utilisateurs, des éléments et des notations, de sorte que tous les algorithmes de recommandation traditionnels puissent être appliqués. L'une de ces techniques est l'approche de découpage contextuelle. [5]

### 4.3.2. Algorithmes d'adaptations

Les algorithmes d'adaptation incorporent directement les contextes dans la fonction de prédiction. Deux sous-catégories sont impliquées : la modélisation indépendante et la modélisation dépendante :

- **modélisation indépendante** : par exemple, TF (Term Frequency ) [25] qui suppose que les contextes sont indépendants des utilisateurs (et des éléments)
- **modélisation dépendante** : exploite les dépendances entre les utilisateurs, les éléments et des contextes, tels que [4], et la méthode linéaire contextuelle parcimonieuse. [46].

Nous incluons également certains algorithmes de recommandation traditionnels dans la ligne de base du package.

Ces algorithmes de recommandation traditionnels ont notamment deux objectifs : D'une part, ces algorithmes peuvent être appliqués après la transformation des données (par exemple, des opérations de scission), étape essentielle des algorithmes de transformation sensibles au contexte. D'un autre côté, il est généralement courant de concurrencer un algorithme de recommandation contextuelle avec des algorithmes non contextuels pour déterminer si l'effet de contexte est significatif ou si un algorithme de recommandation sensible au contexte est nécessaire ou non [2].

## 4.4. Evaluation

La plupart des algorithmes intégrés à CARSKit sont capables d'exécuter la tâche de recommandation suivante : prévision de classement et recommandation d'élément, à l'exception de ceux spécifiquement conçus pour la recommandation top-N. Mais l'évaluation est différente des évaluations traditionnelles, car les contextes sont des intrants

supplémentaires dans le processus d'évaluation.

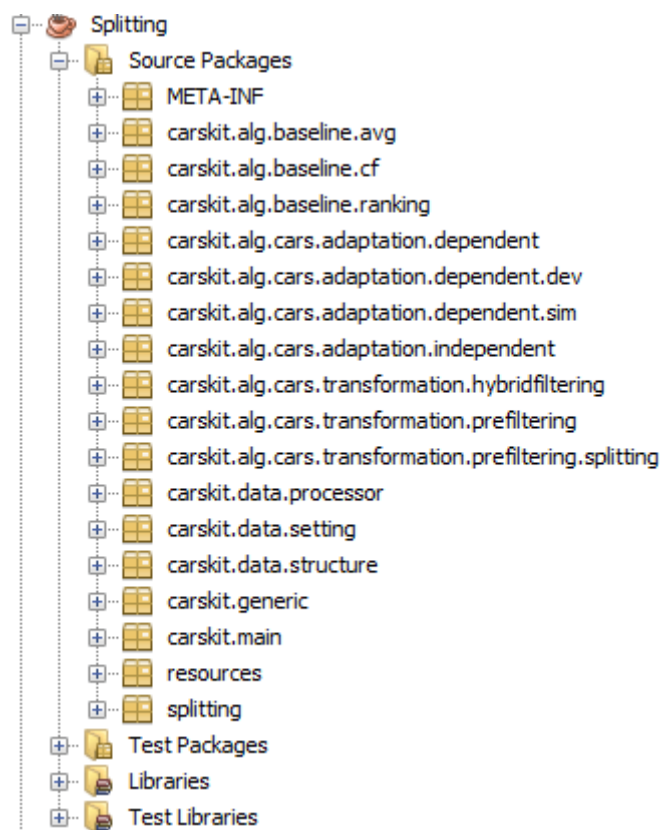
En règle générale, la prédiction de classement peut être évaluée à l'aide de différentes erreurs de prédiction, telles que l'erreur absolue moyenne (MAE), l'erreur quadratique moyenne (RMSE) et l'erreur de prédiction moyenne (MPE). La recommandation d'article peut être évaluée à l'aide de mesures de pertinence, telles que la précision et le rappel, et de mesures de classement, telles que la précision moyenne, le gain cumulé actualisé normalisé et le rang réciproque moyen. [2].

## 4.5. Guide d'utilisation

Dans cette partie, nous allons expliquer les différentes étapes et instructions nécessaires pour utiliser CARSKit.

### 4.5.1. Téléchargement de fichier CARSKit

- **Étape 1 :** Vous devez télécharger le dossier de CARSKit compressé à partir de ce lien : <https://github.com/irecsys/CARSKit/>
- **Étape 2 :** Décompressez le fichier téléchargé dans le projet que vous avez créé sur votre outil





### 4.5.2. Format de donnée

En règle générale, les données de classement contextuelles peuvent être stockées dans deux formats : format libre et format compact, comme indiqué dans la figure 15

(a) Loose Format					(b) Compact Format				
UserID	ItemID	Rating	Context	Condition	UserID	ItemID	Rating	Time	Location
U1	T1	3	Time	Weekend	U1	T1	3	Weekend	Work
U1	T1	3	Location	Work	U2	T2	4	Weekday	Home
U2	T2	4	Time	Weekday	U1	T1	4	Weekend	Home
U2	T2	4	Location	Home	U2	T2	2	Weekday	Work

**Figure 15** : format de donnée [2]

La dimension de contexte est identique à la variable contextuelle, par exemple, Heure et lieu, comme indiqué dans l'exemple ci-dessus. Les conditions de contexte font référence à des valeurs spécifiques dans une dimension, par exemple. Week-end et Weekday sont deux conditions de la dimension Time. Le format libre suppose qu'il n'existe qu'une note pour chaque paire <user, Item> dans les contextes associés, le format compact permettant de stocker plusieurs évaluations pour une même paire <user, Item> dans différentes situations contextuelles. Prenons l'exemple présenté dans la figure 15 ci-dessus. Les deux premières lignes au format non structuré représentent en réalité une notation unique par U1 pour T1 dans les contextes {Week-end, Travail}. En format compact, chaque ligne représente un seul profil d'évaluation contextuelle ; c'est-à-dire qu'il n'y a que deux profils d'évaluation contextuels dans le format standard, mais quatre profils d'évaluation dans le format compact dans cet exemple. [2].

UserID	ItemID	Rating	Time:Weekend	Time:Weekday	Location:Home	Location:Work
U1	T1	3	1	0	0	1
U2	T2	4	0	1	1	0
U1	T1	4	1	0	1	0
U2	T2	2	0	1	0	1

**Figure 16** : format binaire [2]

La plupart des informations contextuelles sont sous la forme de données catégoriques. Les formats libres et compacts augmenteront la pression de stockage et les coûts de calcul. Dans CARSKit, nous stockons la notation contextuelle dans un format binaire, comme indiqué dans la figure 16, capable d'améliorer considérablement les performances de fonctionnement. Pour aider les utilisateurs finaux à préparer les données d'évaluation, nous proposons deux méthodes, TransformationFromLooseToBinary et TransformationFromCompactToBinary, en tant que transformateur de données dans notre boîte à outils. [2].

### 4.5.3. Préparation de donnée

Il est préférable de suivre les étapes ci-dessous pour préparer vos données :

- **Étape 1.** Préparez votre jeu de données au format libre ou Compact, et CARSKit convertira automatiquement vos données au format binaire. Sachez que vous devez ajouter un en-tête à vos données, comme indiqué dans le tableau. Enregistrez votre fichier au format txt ou csv en utilisant une virgule comme séparateur.
- **Étape 2.** Créez un dossier pour chaque ensemble de données et placez le fichier ratings.txt ou ratings.csv dans chaque dossier. Il est suggéré, car CARSKit créera un sous-dossier de travail sous le chemin où vos données de classement sont stockées
- - **Étape 3.** Attribuez le chemin d'accès aux données au fichier de configuration et exécutez tous les algorithmes permettant de tester la transformation des données.

### 4.5.4. Configuration

Vous devez créer une figure setting.conf qui est déjà incluse dans le référentiel CARSKit 3.

Les principales instructions à configuré dans ce fichier sont introduites comme suit :

- **Chemin de données :**

```
# dataset: contextual rating data, or raw rating
dataset.ratings.wins=D:\\1MonProjet\\RecSys\\data\\ratings2.txt
#dataset.ratings.lins=/users/yzheng/desktop/data/restaurant/ratings.txt
```

**Remarque :** pour configuré le dataset sur les plateformes Windows vous devez utiliser le `dataset.ratings.wins`

- **Configuration des rattings :**

```
ratings.setup=-threshold -1 -datatransformation -1
```

**Remarque :** en définissant un seuil de classement (par exemple, le seuil 3), les notations sont converties en notations binaires (par exemple, les notations supérieures à 3 seront égales à 1; sinon, 0), qui est généralement adoptée lors de l'évaluation des métriques de classement. S'il existe déjà des données d'évaluation binaires dans le dossier «CARSKit.Workspace» et que vous n'avez pas besoin de transformation de données, définissez la valeur négative sur `-datatransformation`; sinon, définissez-le comme toute valeur positive, par exemple 1

- **le choix de l’algorithme :**

```
# baseline-Avg recommender: GlobalAvg, UserAvg, ItemAvg, UserItemAvg
# baseline-Context average recommender: ContextAvg, ItemContextAvg, UserContextAvg
# baseline-CF recommender: ItemKNN, UserKNN, SlopeOne, PMF, BPMF, BiasedMF, NMF, SVD++
# baseline-Top-N ranking recommender: SLIM, BPR, RankALS, RankSGD, LRMF
# CARS - splitting approaches: UserSplitting, ItemSplitting, UISplitting; algorithm options: e.g.,
usersplitting -traditional biasedmf -minlenu 3 -minleni 3
# CARS - filtering approaches: SPF, DCR, DCW
# CARS - independent models: CPTF
# CARS - dependent-dev models: CAMF_CI, CAMF_CU, CAMF_C, CAMF_CUCI, CSLIM_C, CSLIM_CI, CSLIM_CU,
CSLIM_CUCI, GCSLIM_CC
# CARS - dependent-sim models: CAMF_ICS, CAMF_LCS, CAMF_MCS, CSLIM_ICS, CSLIM_LCS, CSLIM_MCS,
GCSLIM_ICS, GCSLIM_LCS, GCSLIM_MCS

# recommender=usersplitting -traditional biasedmf -minlenu 2 -minleni 2
recommender=camf_cu
```

- **Configuration des tâches de recommandations et protocoles des évaluations :**

```
# main option: 1. test-set -f test-file-path; 2. cv (cross validation) -k k-folds [-p on, off]
# 3. leave-one-out; 4. given-ratio -r ratio;
# other options: [--rand-seed n] [--test-view all] [--early-stop loss, MAE, RMSE]
# evaluation.setup=cv -k 5 -p on --rand-seed 1 --test-view all --early-stop RMSE
# evaluation.setup=given-ratio -r 0.8 -target r --test-view all --rand-seed 1
# main option: is ranking prediction
# other options: -ignore NumOfPopularItems

evaluation.setup=cv -k 5 -p on --rand-seed 1 --test-view all --early-stop MAE
evaluation.setup=given-ratio -r 0.8
item.ranking=on -topN 10
```

**Remarque :** ce sera une tâche de prédiction d’évaluation si vous désactivez l’élément `item.ranking`; sinon, il s’exécutera en tant que tâche de recommandation parmi les N premiers.

Vous êtes en mesure de choisir la validation croisée du pli k ou une simple évaluation de test de train. Une fois que vous affectez une valeur fixe à k, l’évaluation est effectuée sur les mêmes plis k même si vous examinez des algorithmes de recommandation différents.

- **Configuration des sorties :**

```
# main option: is writing out recommendation results; [--fold-data --measures-only --save-model]
output.setup=-folder CARSKit.Workspace -verbose on, off --to-file results_all_2016.txt
```

En principe, toutes les sorties seront placées dans le dossier nommé «CARSKit.Workspace», créé sous votre chemin de données. Et tous les résultats de l’évaluation seront ajoutés au même fichier «results.txt»

- **Exécution des algorithmes :**

```
java -jar CARSKit.jar -c setting.conf
```

```
java -jar CARSKit.jar -c CAMF.conf PMF.conf UserSplitting.conf
```

Vous pouvez exécuter plusieurs algorithmes ou configurations en ajoutant des fichiers de configuration à la commande ci-dessus.

- **Interprétation des résultats :**

Les sorties fournissent une liste de statistiques simples sur les données, telles que le nombre d'utilisateurs uniques, d'éléments et de dimensions ou de conditions de contexte, ainsi que des statistiques de classification, telles que la moyenne, le mode et la médiane. Dans la figure16, nous avons des résultats pour la tâche de prédiction de notation (évaluée par des erreurs de prédiction) et la tâche de recommandation du top-N (évaluée par précision, rappel, NDCG et MRR), suivies des paramètres en cours afin que les utilisateurs puissent rechercher les meilleures configurations

```

/*****
*****
*
* Dataset: C:\Users\irecs\Desktop\Data\music\CARSKit.Workspace\ratings_binary.txt
* User amount: 42
* Item amount: 139
* Rate amount: 3938
* Context dimensions: 8 (drivingstyle, landscape, mood, naturalphenomena, roadtype,
sleepiness, trafficconditions, weather)
* Context conditions: 34 (drivingstyle: 3, landscape: 5, mood: 5, naturalphenomena: 5,
roadtype: 4, sleepiness: 3, trafficconditions: 4, weather: 5)
* Context situations: 27
* Contextual Data density: 8.0978%
* Scale distribution: [2.0 x 705, 4.0 x 513, 0.0 x 122, 1.0 x 1452, 5.0 x 494, 3.0 x 652]
*
* Average value of all ratings: 2.447647
* Standard deviation of all ratings: 1.474440
* Mode of all rating values: 1.000000
* Median of all rating values: 2.000000
*
*****/
Dataset: ...ARSKit.Workspace\ratings_binary.txt
DataPath: C:\Users\irecs\Desktop\Data\music\CARSKit.Workspace\ratings_binary.txt
Rating data set has been successfully loaded.
With Setup: cv -k 5 -p on --rand-seed 1 --test-view all

Final Results by CAMF_C, MAE: 1.297121, RMSE: 1.469336, NAME: 0.259424, rMAE: 2.447637,
rRMSE: 2.836681, MPE: 0.000000, numFactors: 10, numIter: 120, lrate: 2.0E-10, maxlrate: -1.0,
regB: 0.001, regU: 0.001, regI: 0.001, regC: 0.001, isBoldDriver: true, Time: '00:01','00:00'

Final Results by BiasedMF, Pre5: 0.050361, Pre10: 0.025180, Rec5: 0.149967, Rec10: 0.149967,
AUC: 0.589942, MAP: 0.084189, NDCG: 0.110593, MRR: 0.122737, numFactors: 10, numIter: 100,
lrate: 2.0E-7, maxlrate: -1.0, regB: 0.001, regU: 0.001, regI: 0.001, regC: 0.001,
isBoldDriver: true, Time: '00:00','00:00'

```

**Figure 17 : Résultat**

## **4.5. Conclusion**

Ce chapitre avait pour objectif de vous faire part de l'utilité que vous pouvez apporter à la bibliothèque CARSKit sur les systèmes de recommandation sensibles aux contextes. Nous avons présenté les différentes étapes, ainsi que les instructions nécessaires pour l'utilisation et la configuration de cette dernière.

# *CONCLUSION GENERALE*

## Conclusion générale

La personnalisation de l'information émerge comme une approche capitale dans le développement des systèmes du futur. Parmi ces derniers, nous pouvons citer les systèmes de recommandation qui donnent de l'importance aux préférences des utilisateurs, dans le but de leur proposer des ressources à acheter ou à consulter répondent au mieux à leurs besoins.

Ils sont devenus des efficaces dans le commerce électronique, la recherche documentaire, le tourisme, etc., en fournissant des suggestions pertinentes au sein d'une grande masse d'informations. L'intégration du contexte dans les systèmes de recommandation classiques, permet d'améliorer les performances de ces derniers.

Dans notre travail, nous avons mis en place une application de recommandation basée sur l'approche d'intégration de contexte Pré-filtrage contextuelle, La particularité de cette approche est qu'elle se fait avant que la méthode principale de recommandation soit lancée, en utilisant l'une des techniques de cette approche Item Splitting

En vue de confirmer l'efficacité de notre application, nous avons effectué des tests, d'évaluation en utilisant le protocole de hold out.

De cela, nous pouvons conclure que l'intégration du contexte dans les systèmes de recommandation, permet d'améliorer les performances de ces derniers, en fournissant des informations supplémentaires qui vont être exploitées afin d'effectuer de meilleures recommandations, et de ce fait, dire que la contextualisation a un impact positif sur le processus de recommandation.

Comme perspectives nous envisageons apporter quelques améliorations à savoir :

Implémenter les autres algorithmes de l'approche de découpage à savoir par utilisateur et celui par item utilisateur et ceux pour pouvoir intégrer d'autres types d'informations comme information social et voir l'impact sur taux de prédiction.

# *Références Bibliographique*

- [1] Adomavicius, G. and Tuzhilin, A. (2005). Toward the next generation of recommender systems : A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6) :pages 734-749.
- [2] A User's Guide to CARSKit Center for Web Intelligence, School of Computing, DePaul University 243 South Wabash Ave, Chicago, Illinois, USA 60604 .2015 .
- [3] [L. Baltrunas, X. Amatriain, 2009] Baltrunas. L, Amatriain. X, Towards Time-Dependant Recommendation based on Implicit Feedback, cars 2009.
- [4] Baltrunas. L, Ludwig, B. and Ricci, F. Matrix factorization techniques for context aware recommendation. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 301–304. ACM, 2011.
- [5] Baltrunas. L. and Ricci, F. Experimental evaluation of context-dependent collaborative filtering using item splitting. *User Modeling and User-Adapted Interaction*, pages 1–28, 2009.
- [6] Basu, C., Hirsh, H., Cohen, W. and Manning, N. C. Technical paper recommendation : A study in combining multiple information sources, 2001.
- [7] Bettini C., Brdiczka O., Henriksen K., Indulska J., Nicklas D., Ranganathan A., and Riboni D., A survey of context modelling and reasoning techniques. *Journal of Pervasive and Mobile Computing*, 2010.
- [8] Bill N. Schilit and Marvin M. Theimer. Disseminating active map information to mobile hosts. *Network, IEEE*, 8(5):22-32, Sep/Oct 1994.
- [9] Brown, P. The stick-e document: a framework for creating context-aware applications. pages 182–196, 1996.
- [10] Burke, R. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4) :331–370, 2002.



- [11] Castagnos, S. « Modélisation de comportements et apprentissage stochastique non supervisé de stratégies d'interactions sociales au sein de systèmes temps réel de recherche et d'accès à l'information », thèse de doctorat de l'université Nancy 2. Novembre 2008.
- [12] Chaari, T., Laforest, F., and Flory, A., Adaptation des applications au contexte en utilisant les services web : Le projet secas. In La Deuxième Journées Francophones Mobilité et Ubiquit, 2005.
- [13] Chaari, T. Adaptation d'applications pervasives dans des environnements multi-contextes. Thèse de doctorat, INSA de Lyon, 2007.
- [14] Daoud. M, Recherche contextuelle d'information, RJCRI 2007.
- [15] Dey, A. Providing Architectural Support for Building Context-aware Applications. PhD thesis, Atlanta, GA, USA, 2000.
- [16] Dey, K. A., Abowd, D. G., Towards a better understanding of context and context awareness. In Computer Human Interactions (CHI2000) Workshop on the What, Who, Where and How of Context Awareness. 2000.
- [17] Dey, A., Abowd, G., and Salber, D. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human- Computer Interaction*, 16(2) :97–166, 2001
- [18] Erdt, M., Fernandez, A., and Rensing, C. Evaluating recommender systems for technology enhanced learning: a quantitative survey. *IEEE Transactions on Learning Technologies*, 8(4), (pp. 326-344), 2015.
- [19] Gabrielsson, S. & Gabrielsson, S. The use of Self-Organizing Maps in Recommender Systems, A survey of the Recommender Systems field and a presentation of a State of the Art Highly Interactive Visual Movie Recommender System. Mémoire de master, Uppsala University, 2006.
- [20] Ghayam, Y. E. La Sensibilité au Contexte dans un Environnement Mobile. PhD thesis, Université Mohammed V Souissi-RABAT, Ecole Nationale Supérieure d'Informatique et d'Analyse des Systèmes, 2001.
- [21] Goldberg, D., Nichols, D., Oki, M., and Terry, D. (1992). Using collaborative filtering to weave an information tapestry. In *Commun. ACM*, volume 35, pages 61-70

- [22] [Henricksen. K et Indulska. J,2006] Henricksen K. and Indulska J., Developing context-aware pervasive computing applications: Models and approach. *Journal of Pervasive and Mobile Computing*, 2006.
- [23] Herlocker, J. L., Konstan, J. A., Terveen, L. G., & Riedl, J. T. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1), (pp. 5-53), 2004.
- [24]Jannach, Zanker, Felfernig and Friedrich, *Recommender Systems –2010*
- [25] Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver. Multiverse recommendation: n-dimensional tensor factorization for context aware collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 79–86. ACM, 2010.
- [26] Laouar, A. Korichi, K. Un système de recommandation pour l’assistance à la navigation sur Internet. Mémoire MASTER ACADEMIQUE Spécialité : Informatique industrielle. UNIVERSITE KASDI MERBAH OUARGLA. 2016.
- [28] Lemdan Roza. Thèse de Doctorat de l’Université Paris-Saclay préparée à Ecole doctorale n°580 Sciences et technologies de l’information et de la communication, 2016.
- [29] Malone, T., Brobst, S., Cohen, S., Grant, K., and Turbak, F. (1987). Intelligent information des systèmes de partage. In *Communications of the ACM*, volume 30, pages 390-402.
- [30] Manouselis, N., Drachsler, H., Verbert, K., and Duval, E. *Recommender Systems for Learning*. Berlin, Springer, 2012.
- [31] Margaritis, K. and Vozalis, E. Analysis of recommender systems’ algorithms. In *6th Hellenic European Conference on Computer Mathematics its Applications (HERCMA)*, Athens, Greece, 2003.
- [32] Michael J. Pazzani and Daniel Billsus. *The adaptive web*. chapter Content-based Recommendation Systems, pages 325-341. Springer Verlag, Berlin, Heidelberg, 2007.
- [33] Michael D. Ekstrand, John T. Riedl, Joseph A. Konstan. (2010). *Collaborative Filtering Recommender Systems*, Foundations and Trends R in Human Computer Interaction, Boston-Delft.
- [34] Panniello. U, Tuzhilin. A, Gorgoglione. M, *Experimental Comparison of Pre vs. Post-filtering Approaches in Context Aware Recommender Systems*, 2009.

- [35] Pascoe, J. Adding generic contextual capabilities to wearable computers. In (ISWC, pages 92–99, 1998.
- [36] Rey, G., Coutaz, J., Le Contexteur : une abstraction logicielle pour la réalisation de systèmes interactifs sensibles au contexte. In Proceedings of Interaction Homme Machine (IHM2002), 2003.
- [37] Ryan N. S, Pascoe J. and Morse D. R., Enhanced reality fieldwork: the context-aware archeological assistant. Gaffney, Leusen and Exxon edition, 1997.
- [38] Said, E. W. De Luca, and S. Albayrak. Inferring contextual user profiles – improving recommender performance. In ACM RecSys’11, Workshop on Context-Aware Recommender Systems, 2011.
- [39] Schilit, B., Adams, N., and Want, R. Context-aware computing applications. In The 1994 First Workshop on Mobile Computing Systems and Applications, pages 85–90, Washington, DC, USA. IEEE Computer Society, 1994.
- [40] Schilit, B., Hilbert, D., and Trevor, J. Context-aware communication. IEEE Wireless Commun, 9(5):46–54, 2002.
- [41] Shani, G., & Gunawardana, A. Evaluating recommendation systems. In Recommender systems handbook (pp. 257-297). Springer, Boston, MA, 2011.
- [42] SOLTANI Réda, Impact de la prise en compte d’information contextuelle sur la pertinence des systèmes de recommandations, mémoire de master, Université d’Oran, 2011.
- [43] Soualah-Alila, F., CAMLearn: Une Architecture de Système de Recommandation Sémantique Sensible au Contexte. Application au Domaine du MLearning. Informatique. Université de bourgogne. 2015.
- [44] Tadlaoui. M. Système de recommandation de ressources pédagogiques fondé sur les liens sociaux : formalisation et évaluation. THESE de DOCTORAT DE L’UNIVERSITE DE LYON opérée au sein de L’INSA Lyon et délivrée en partenariat international avec L’université de Tlemcen, 2018.
- [45] Weiser, M. The computer for the 21st century. Scientific American Special Issue on Communications, Computers, and Networks, 1991.

[46] Zheng. Y, Mobasher. B and Burke. R. Deviation based contextual SLIM recommenders. In Proceedings of the 23rd ACM Conference on Information and Knowledge Management, pages 271–280, 2014

[47] Zheng. Y, Mobasher. B and Burke. R. Carskit: A java based context aware recommendation engine. In Proceedings of the 15th IEEE International Conference on Data Mining Workshops. IEEE, 2015.

[43] [Weiser. M, 1991] Weiser, M. The computer for the 21st century. Scientific American Special Issue on Communications, Computers, and Networks,1991.

[44] Zheng. Y, Mobasher. B and Burke. R. CSLIM: Contextual SLIM recommendation algorithms. In Proceedings of the 8th ACM Conference on Recommender Systems, pages 301– 304. ACM, 2014.

[45] Zheng. Y, Mobasher. B and Burke. R. Deviation based contextual SLIM recommenders. In Proceedings of the 23rd ACM Conference on Information and Knowledge Management, pages 271–280, 2014

[46] Zheng. Y, Mobasher. B and Burke. R. Carskit: A java based context aware recommendation engine. In Proceedings of the 15th IEEE International Conference on Data Mining Workshops. IEEE, 2015.

# *Références Webographie*

[48] [Amazon, 2019] Amazon : [www.amazon.com](http://www.amazon.com) consulté le 01/05/2019

# *Liste des Abréviations*

<b>Acronyme</b>	<b>Signification</b>
CARS	système de recommandation sensible au contexte.
TF	Terme fequency
RMSE	root mean squard error
MPE	mean percentage erroe
NDCG	Normalised Discount Cumulative Gain
ACP	Analyse en Composant Principale
SR	Système de recommandation
FC	Système de recommandation
FCB	filtrage basé conteneue
EPF	Exact Pre-Filtering
UI	User and Item
KNN	K nearest Neighborhood

## Résumé

Les systèmes de recommandation (SR) classique offre une solution à la quantité croissante du contenu produit chaque jour. Ils fournissent aux utilisateurs des contenus qui sont les plus pertinents pour eux, mais en négligeant le contexte courant utilisateur. D'où l'apparition des systèmes de recommandation sensible au contexte « CARS » qui intègre l'information contextuelle dans une recommandation.

Dans notre projet de fin d'étude, nous visons à implémenter l'une des approches les plus efficaces dans les CARS à savoir : approche de découpage. Ainsi nous explorons la bibliothèque CARSKits implémentant la majorité des algorithmes d'état de l'art des CARS. CARSKits est très utile pour l'évaluation des futures algorithmes de recommandation proposés et plus particulièrement ceux sensible au contexte.

**Mots-clés :** *Information, Système de recommandation, pré-filtrage contextuelle, Sensibilité au contexte, approche de découpage.*

## Abstract

Conventional Recommendation Systems (SR) offer a solution to the increasing amount of content produced each day. They provide users with content that is most relevant to them, but neglecting the current user context. Hence the emergence of context-aware recommendation systems "CARS" that integrates contextual information into a recommendation.

In our end-of-study project, we aim to implement one of the most effective approaches in the CARS namely: Splitting approach. Thus, we explore the CARSKits library implementing the majority of state-of-the-art CARS algorithms. CARSKits is very useful for the evaluation of future recommendation algorithms proposed and more particularly those sensitive to the context.

**Keywords:** *Information, Recommendation System, Contextual Pre-filtering, Context Aware, Splitting Approach.*

## ملخص

تقدم أنظمة التوصية التقليدية (SR) حلاً للكمية المتزايدة من المحتوى المنتج كل يوم. أنها توفر للمستخدمين المحتوى الأكثر صلة بهم، ولكن مع إهمال سياق المستخدم الحالي. ومن هنا نشأت أنظمة توصية حساسة للسياق (CARS) تدمج المعلومات السياقية في توصية.

في مشروعنا النهائي للدراسة، نهدف إلى تنفيذ أحد الأساليب الأكثر فاعلية في CARS وهي: نهج القطع. وبالتالي، فإننا نستكشف مكتبة CARSKit التي تنفذ غالبية خوارزميات CARS الحديثة. تعد مجموعات CARSKit مفيدة جداً لتقييم خوارزميات التوصيات المستقبلية المقترحة، وبشكل خاص تلك الحساسة للسياق.

**الكلمات المفتاحية:** المعلومات، نظام التوصيات، التصفية المسبقة للسياق، حساسية السياق، نهج القطع.