Système de recommandations utilisant une combinaison de filtrage collaboratif et de segmentation pour des données implicites

par

Simon Renaud-Deputter

Mémoire présenté au Département d'informatique
en vue de l'obtention du grade de maître ès sciences (M.Sc.)

FACULTÉ DES SCIENCES
UNIVERSITÉ DE SHERBROOKE

Sherbrooke, Québec, Canada, juin 2013

Your file Votre référence
ISBN: 978-0-494-95122-4

Our file Notre référence
ISBN: 978-0-494-95122-4

Le 19 juin 2013

le jury a accepté le mémoire de Monsieur Simon Renaud-Deputter dans sa version finale.

Membres du jury

Professeur Shengrui Wang
Directeur de recherche
Département d'informatique

Professeur André Mayers
Évaluateur interne
Département d'informatique

Professeur Gabriel Girard
Président rapporteur
Département d'informatique

SOMMAIRE

Avec la montée de la technologie et la facilité d'accès à Internet, les utilisateurs sont submergés par un large éventail de choix disponibles et une quantité considérable d'informations [6]. Il devient nécessaire d'avoir accès à des outils et des techniques efficaces pour filtrer les données et de les rendre utilisables pour des opérations de tous les jours. À cette fin, des systèmes de recommandations, qui ont fait l'objet de recherche active et de développement au cours des 15 dernières années, sont maintenant capables de fournir aux utilisateurs des choix [51] sur ce qu'ils aimeraient lire, acheter, regarder, manger, etc.

La problématique étudiée dans ce mémoire est l'utilisation d'informations implicites pour construire des systèmes de recommandations en utilisant une approche par filtrage collaboratif. Beaucoup de travaux ont été faits sur l'utilisation de filtrage collaboratif à l'aide d'informations explicites telles que les cotes [48], [43], [19], [33]. Cependant, les techniques développées pour les systèmes de recommandations comprenant des articles sans informations explicites restent rudimentaires. Le plus grand défi vis-à-vis les systèmes de recommandations à informations implicites est l'absence de rétroaction de la part de l'utilisateur si nous n'utilisons pas un expert comme par exemple, un vendeur. En outre, comme il est mentionné dans [51], lorsque qu'un système avec cote existe, la proportion des éléments évalués est souvent inférieure à 1%. Par conséquent, même pour les systèmes de recommandations qui utilisent des informations explicites telles que les cotes, il est

crucial d'avoir une méthode qui tire profit des informations implicites.

Les progrès dans ce domaine sont timides depuis les dernières années. Il y a eu des études sur les recommandations par rapport aux médias sociaux en se basant sur des utilisateurs et des mots-clés [18], la modélisation probabiliste [30] et la modélisation sémantique basée sur la recommandation de nouvelles [29]. S'il est vrai que ces techniques utilisent des informations implicites, seuls quelques-uns [40], [23] abordent la question de recommander des produits d'un magasin sans l'utilisation d'informations explicites. Ces méthodes nécessitent généralement la disponibilité d'un expert afin de prendre la rétroaction d'un client ou le réglage de nombreux paramètres.

Dans notre étude, nous avons réussi à élaborer un algorithme permettant de soumettre des recommandations personnelles à un utilisateur en utilisant seulement des informations implicites. Notre technique, lorsque comparée à un système semblable qui utiliserait des cotes comme informations explicites, génère de très bons résultats. De plus, lorsque la méthode développée est comparée à d'autres systèmes utilisant de l'information implicite, elle offre des résultats qui sont comparables et parfois supérieurs à ceux-ci.

Mots-clés : système de recommandations ; segmentation ; information implicite ; filtrage collaboratif ; division hiérarchique.

REMERCIEMENTS

Je veux d'abord remercier mon directeur de recherche Shengrui Wang pour sa motivation et ses idées apportées au cours de mes recherches.

Je remercie Tengke Xiong, auteur de l'algorithme de segmentation DHCC, d'avoir répondu à mes questions sur les subtilités de la segmentation hiérarchique.

Je remercie les membres du laboratoire Prospectus qui ont su m'écouter et échanger avec moi à propos de divers sujets.

Je remercie mes parents qui ont toujours été là pour me soutenir au cours de ma maîtrise.

Je remercie l'Université de Sherbrooke pour son soutien au cours de mes études universitaires.

TABLE DES MATIÈRES

SOMMAIRE ii

REMERCIEMENTS iv

TABLE DES MATIÈRES v

LISTE DES TABLEAUX viii

LISTE DES FIGURES ix

INTRODUCTION 1

CHAPITRE 1 — Système de recommandations 5

1.1 Types d'approches 6

1.1.1 Filtrage collaboratif 6

1.1.2 Réseau de confiance 9

1.1.3 Règles d'association 11

1.1.4 Modèle latent 12

1.2 Validation 14

1.2.1 La racine de l'erreur quadratique moyenne 16

1.2.2 Précision, rappel et mesure-f 16

1.2.3 Courbe ROC 18

1.3 Défis généraux 19

1.3.1 Démarrage à froid 19

1.3.2 Données dispersées 21

1.3.3 Défis techniques 22

1.3.4 Tendance en fonction du temps 22

1.4 Défis de l'information implicite 23

1.4.1 L'absence de commentaires des utilisateurs 24

1.4.2 Cohérence des données 24

1.4.3 Confiance vs préférence 25

1.4.4 Besoin de redéfinir les métriques d'évaluation 25

1.5 Besoin d'une technique de segmentation 26

CHAPITRE 2 — Segmentation **27**

2.1 Type d'attributs 30

2.2 Techniques de segmentation avec données catégoriques 32

2.2.1 ROCK 33

2.2.2 CACTUS 34
2.2.3 COOLCAT 34
2.2.4 AT-DC 35
2.3 DHCC 36
2.3.1 Fonctionnement de l'algorithme DHCC 38
2.3.2 Avantages de DHCC 40
2.3.3 Limites 42
2.4 Segmentation comme outil de système de recommandations 43

CHAPITRE 3 — Méthodologie et Résultats **45**

3.1 Segmentation 46
3.2 Recommendations 47
3.3 Résultats Expérimentaux 50
3.3.1 Validation de l'Hypothèse 50
3.3.2 Validation des recommandations 53
3.3.3 Comparaison entre divers systèmes de recommandations 57

CONCLUSION ET PERSPECTIVES **61**

BIBLIOGRAPHIE **64**

LISTE DES TABLEAUX

2.1 Exemple d'une matrice indicatrice 37

3.1 Exemple d'une matrice d'entrée 46

3.2 Résumé des notations 48

3.3 Ratio global des utilisateurs ayant une cote définie par le seuil t 53

3.4 Pourcentage des utilisateurs dont les cotes pour les recommandations sont définies par le seuil t 56

3.5 Précision des recommandations définies par le seuil t pour des systèmes de recommandations utilisant des informations implicites, avec $m = 10$. . . . 59

LISTE DES FIGURES

1.1 Courbe ROC 19

3.1 Résultat d'une segmentation par DHCC. 47

INTRODUCTION

Avec la facilité d'accès à Internet, nous sommes de plus en plus exposés à une multitude d'informations. Avec tous les blogues, les journaux, les magasins et bien d'autres, cela apporte beaucoup de diversités aux utilisateurs. Toutefois, avec cette multitude de sources d'informations, la surcharge d'information peut devenir problématique. Afin de remédier à ce problème, divers outils ont été développé pour filtrer les informations avant de les transmettre aux utilisateurs. Les systèmes de recommandations sont des outils permettant un tel filtrage. Le but principal de ces systèmes est de faciliter la prise de décisions pour les utilisateurs en leur offrant des informations selon leurs préférences. Il existe déjà des systèmes de recommandations pour divers produits tels que des films (Netflix), de la musique (last.fm)et des livres (Amazon).

Il y a principalement deux grandes catégories de systèmes de recommandations. L'une se fait avec un filtrage basé sur le contenu et l'autre se fait avec un filtrage collaboratif [29]. L'approche basée sur le contenu est mise en oeuvre en prenant en compte les caractéristiques des produits recommandés et en créant des groupes de produits en utilisant une mesure de similitude sur leur contenu. Par contenu, on sous-entend une relation entre un mot et un produit. L'un des principaux inconvénients de cette approche est que les caractéristiques sont généralement acquises à l'aide des informations externes qui ne sont pas toujours disponibles. Dans un autre ordre d'idées, le filtrage collaboratif utilise géné-

ralement un voisinage d'utilisateurs similaires et les produits recommandés en fonction de l'historique des autres utilisateurs au sein du même voisinage. Lorsque les systèmes émettent leurs recommandations, ils peuvent le faire globalement ou localement. «Globalement» signifie que tous les utilisateurs recevront les mêmes recommandations. Alors que «localement» signifie que les éléments recommandés ne seront pas les mêmes pour tous les utilisateurs.

Il y a deux approches dans la littérature lors de l'élaboration d'un système de recommandations utilisant le filtrage collaboratif [48]. La première approche est basée sur la mémoire. C'est-à-dire que le système va émettre ses recommandations en se basant sur un voisinage contenu en mémoire. Alors que la seconde approche est plutôt basée sur un modèle. Ainsi, il faut d'abord créer des modèles qui ressemblent aux comportements des utilisateurs et ensuite, utiliser ces modèles afin d'émettre les recommandations. En pratique, il a été démontré [48] que l'approche par mémoire offre de meilleures performances en terme de précision alors que l'approche par modèle est plus efficace à grande échelle avec de gros ensembles de données.

Il y a principalement deux types d'informations qui sont utilisées par les systèmes de recommandations, l'information implicite et explicite. L'information explicite se traduit comme étant de l'information fournie par l'utilisateur. Il est possible de fournir cette information par l'usage de cotes, de listes de souhaits, de commentaires, etc. Contrairement à l'information explicite, l'information implicite utilise seulement de l'information pour laquelle l'utilisateur n'a pas été sollicité. Par exemple, lorsqu'un utilisateur achète un produit, cela crée un historique d'achat pour cet utilisateur. Or, il est possible pour des systèmes de recommandations à informations implicites d'utiliser ces informations afin de créer différents profils d'utilisateurs.

L'une des principales difficultés avec l'information implicite par rapport à l'information explicite est l'absence d'information en retour sur les intérêts des utilisateurs. Lorsqu'un

utilisateur achète un produit, lit un livre ou une nouvelle, on ne sait pas si cet utilisateur aime ou n'aime pas ce qu'il a consommé. Les seules données bien définies et observables sont des occurrences de motifs répétés, par exemple un même ensemble d'achats présent à des intervalles réguliers. En étudiant ces motifs, il est possible de définir une structure comportementale pour un seul ou un groupe d'utilisateurs qui peut ensuite être utilisée pour créer des recommandations fiables.

Des techniques de segmentation permettent de découvrir des groupes d'utilisateurs significatifs où chaque groupe a une structure comportementale différente et les utilisateurs au sein d'un tel groupe ont des goûts et des préférences similaires. La structure comportementale, c'est-à-dire les préférences face à certains produits, d'un groupe d'utilisateurs est utilisée pour faire des recommandations aux utilisateurs à l'intérieur de ce groupe. Cependant, la découverte de ces groupes et des structures comportementales pour ceux-ci est un très grand défi, car les données sont de très grandes dimensions (il y a un très grand nombre de produits qui peuvent être recommandés) et les structures comportementales sont définies dans des sous-espaces (un groupe d'utilisateurs préfère seulement une petite partie de l'ensemble total des produits).

La technique que nous proposons utilise une méthode de segmentation par division hiérarchique qui est très bien adaptée pour les grandes dimensions et qui ne demande pas de paramètre. Cette technique permet de découvrir les relations entre les utilisateurs en n'utilisant que des informations implicites basées sur les historiques d'achats des utilisateurs. La technique utilise la factorisation de matrice afin de découvrir des informations latentes pour trouver des similitudes entre les utilisateurs. La technique proposée est validée sur un ensemble de données bien connu à savoir MovieLens.

Le reste du mémoire est divisé comme suit : le chapitre 1 présente les enjeux suite à l'utilisation de données implicites, les travaux connexes et les techniques pour établir un système de recommandations. Dans le chapitre 2, nous présentons les approches de

segmentation pour des données catégoriques. Dans le chapitre 3, nous décrivons notre système de recommandations et expliquons comment nous avons adapté notre technique de segmentation pour les systèmes de recommandations. Nous terminons avec une discussion sur les travaux futurs pour améliorer le système de recommandations.

CHAPITRE 1

Système de recommandations

Bien qu'il y ait plusieurs types de systèmes de recommandations, lors de l'élaboration d'un tel système, la première étape est de déterminer quels types de données seront utilisés. Un système à information explicite va utiliser des données pour lesquelles les utilisateurs vont avoir indiqué leurs intérêts par rapport aux produits. Il y a plusieurs façons pour un utilisateur d'exprimer ses préférences pour un produit. Il peut le faire en utilisant une cote, qui est un barème avec une échelle prédéfinie et est souvent utilisée pour la vente de produits en ligne. Aussi, sur certains sites Web, il est possible de s'enregistrer afin de créer un profil et de choisir parmi une liste de préférences, celles qui se rapprochent davantage à nos préférences personnelles. Dans certains cas, en plus de créer un profil, il est possible de créer une liste de souhaits. Cette liste a pour but d'indiquer des produits qui intéressent l'utilisateur, mais qu'il n'a pas encore achetés.

Dans un autre ordre d'idée, un système à informations implicites va utiliser diverses sources d'informations sans avoir à demander quoi que ce soit à l'utilisateur. Un exemple d'information implicite très populaire est l'historique des transactions. D'autres informations comme le temps passé par un utilisateur sur la page descriptive d'un produit,

l'historique des produits visités et le temps passé sur certaines pages peuvent toutes être utilisées afin de rendre le système plus performant.

1.1 Types d'approches

Il y a plusieurs types d'approches pour mettre en place un système de recommandations. Les approches présentées dans cette section s'appliquent aussi bien aux systèmes de recommandations utilisant de l'information explicite qu'implicite. Formellement, un systèmes de recommandations est un outil de filtrage de données où il y a un ensemble d'utilisateurs U, un ensemble de produits P et un fonction utilitaire h qui dénote l'intérêt d'un utilisateur pour un produit. Ainsi, il est possible de voir les systèmes de recommandations comme étant une fonction à maximiser (eq. 1.1)[1].

$$\forall u \in U, i'_u = \arg\max_{i \in I} h(u, i) \tag{1.1}$$

I représente un ensemble de produits pour U et la fonction utilitaire h est particulière au type d'approche qui est utilisée. Aussi, cette fonction varie selon le type d'information utilisé, soit implicite et explicite.

1.1.1 Filtrage collaboratif

Le filtrage collaboratif est une technique qui utilise la collaboration d'éléments (comme des utilisateurs) afin d'effectuer une tâche, comme par exemple émettre une recommandation. Le fonctionnement du filtrage collaboratif va comme suit. Tout d'abord, le système recueille des informations (comme des cotes ou un historique d'achats) à propos des utilisateurs afin d'obtenir leurs préférences. Ensuite, le système va comparer les informations

recueillies avec celles des autres utilisateurs et va trouver ceux qui sont les plus similaires. Lorsque les utilisateurs similaires sont trouvés, le système va recommander des produits (que les utilisateurs similaires ont bien cotés, mais que l'utilisateur ciblé n'a pas acheté) à l'utilisateur. L'hypothèse principale derrière le filtrage collaboratif est que si un utilisateur apprécie un produit, alors les utilisateurs ayant des goûts similaires à cet utilisateur apprécieront aussi ce produit. Par exemple, si un utilisateur A partage les mêmes préférences que l'utilisateur B et que A achète un livre sur Amazon et lui donne une bonne cote, alors ce livre sera recommandé à B.

Le filtrage collaboratif peut se faire autant en gardant toutes les informations relatives aux usagers similaires en mémoire, ou encore en créant un modèle de préférences à partir de ces informations. Avec un système où les informations sont gardées en mémoire, les deux mesures de similarité les plus populaires sont la mesure du cosinus (eq. 1.2)(utilisée ici avec deux vecteurs d'achats, $\vec{x}$ et $\vec{y}$) et la corrélation de Pearson (eq. 1.3)(où les vecteur x et y représentent les cotes attribuées aux produits par les utilisateurs X et Y, et $\bar{x}$ et $\bar{y}$ représentent les moyennes de leurs cotes) [20]. La corrélation de Pearson est utile lorsque le système de recommandations utilise de l'information explicite alors que la mesure du cosinus est utilisée lorsqu'il y a de l'information implicite comme un historique d'achats. Lorsque le filtrage collaboratif est utilisé dans le but de créer un modèle, d'autres techniques comme la segmentation sont utilisées afin de découvrir les informations latentes qui pourraient expliquer l'achat d'un produit ou le résultat d'une cote.

$$cos(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{||\vec{x}|| * ||\vec{y}||} \tag{1.2}$$

$$Corrélation\ de\ Pearson(x, y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{n * stdev(x) * stdev(y)} \tag{1.3}$$

L'une des méthodes les plus populaires de filtrage collaboratif est l'algorithme des k plus proche voisin (kNN). kNN est un algorithme de classification où k est un paramètre qui spécifie le nombre de voisins utilisés. Pour utiliser kNN, le système doit avoir une mesure de similitude pour faire la distinction entre des utilisateurs qui sont près et ceux qui sont éloignés. Selon la problématique étudiée, cette mesure de similitude peut être issue de la distance euclidienne, la mesure du cosinus, la corrélation de Pearson, etc. Ensuite, lorsque le système est interrogé sur une nouvelle information, par exemple une recherche de produits qui pourraient plaire à un individu, il va aller trouver les k utilisateurs qui se trouvent le plus près de l'utilisateur cible. Finalement, un vote majoritaire se fait pour décider quel produit sera recommandé. Par exemple, dans le cas d'un système à information explicite, le système pourrait recommander le produit ayant la meilleure cote. Alors que pour l'information implicite, le système pourrait recommander le produit le plus populaire au sein du voisinage.

Desrosier et al.[12] a fait une étude détaillée des méthodes qui utilisent des voisinages pour faire des recommandations autant pour la prédiction de cotes que pour la prédiction de produits. Gantner et al.[15] a mis en oeuvre un système pour faire les recommandations en utilisant kNN où le regroupement des données peut se faire autant en se basant sur les similitudes entres les utilisateurs qu'en se basant sur les similitudes entre les produits. Dans le but d'estimer des similitudes entre les utilisateurs ou entre les articles, ils utilisent la mesure de cosinus. L'un des problèmes à l'utilisation de kNN est que le voisinage proposé est basé sur un nombre fixe de voisins. Ainsi, si le nombre de voisins réellement similaires à un utilisateur est inférieur à k, ce dernier se verra offrir des recommandations qui pourraient ne pas correspondre à ses préférences.

Dans [40], les auteurs créent un système de recommandations basé sur le filtrage collaboratif. Ils utilisent des informations implicites en provenance d'un magasin qui vend des produits de rénovation domiciliaire. Ils commencent par regrouper des articles dans des

catégories telles que « tuyau ». Par la suite, pour chaque type de tuyaux, le produit est traité comme un tuyau générique. Le système de recommandations développé est destiné à être utilisé par un expert (dans ce cas, un vendeur) afin de suggérer les produits appropriés basés sur la catégorie prescrite par le système. Parce que ce système de recommandations est destiné à être utilisé par un expert et suggère qu'une seule catégorie, on ne peut pas appliquer ce système directement à un domaine d'historique d'achats où il est important de recommander à un utilisateur un produit spécifique sans avoir à utiliser un expert.

1.1.2 Réseau de confiance

Les réseaux de confiances utilisent des principes similaires au filtrage collaboratif, mais poussent plus loin la similarité entre les utilisateurs. Les recherches sur les réseaux de confiances sont motivées par l'hypothèse qu'il existe d'autres facteurs, que les mesures de similarité traditionnelles utilisées dans le filtrage collaboratif, qui peuvent pousser un utilisateur à acheter un produit [36]. Cette hypothèse se base sur l'idée que dans la vie de tous les jours, les gens demandent parfois conseils à des personnes réputées, discutent d'un produit avec leurs amis, etc. Contrairement au filtrage collaboratif où la similarité se fait par rapport aux profils des individus, l'intérêt principal des réseaux de confiance est la relation de confiance qui relie les utilisateurs entre eux.

Le principal avantage d'ajouter un réseau de confiance à un système utilisant le filtrage collaboratif est que celui-ci permet d'assouplir le problème du démarrage à froid. Avec le réseau de confiance, même si l'utilisateur est nouveau dans le système, il pourra quand même recevoir des recommandations s'il est relié à d'autre individu. Évidemment, cela nécessite que l'individu connaisse d'autres utilisateurs qui font partie de ce système. Aussi, lorsqu'un produit n'est pas trouvé dans le réseau immédiat de l'utilisateur, la recherche

peut aller de l'avant avec un second degré de confiance et ainsi de suite. Toutefois, même si cela offre une très grande couverture, plus la recherche s'éloigne de la source, plus la précision diminue.

Comme il est difficile de quantifier la confiance, certains chercheurs [34] ont tenté de formaliser la confiance entre différents individus. Il y a deux principales catégories de confiance : la confiance interpersonnelle propre à un contexte et la confiance impersonnelle. La première indique une confiance où un utilisateur fait confiance à son entourage seulement pour un contexte bien précis et peut faire confiance à un tout autre groupe pour un différent contexte. Par exemple, un utilisateur pourrait faire confiance à son entourage pour aller voir un film, mais pourrait aussi faire confiance à un agent d'immeuble pour faire l'achat d'une maison. La seconde indique une confiance systématique qui décrit la confiance d'un utilisateur comme étant un tout à l'intérieur d'un même groupe. Par exemple, deux individus se font confiance car ils font partie du même groupe.

Jamali et al.[25] propose un système qui, en plus d'utiliser un filtrage collaboratif avec de l'information explicite, utilise un réseau de confiance. Le système proposé permet de recommander des produits à un utilisateur en se basant sur les cotes de son réseau de confiance. Afin d'évaluer un produit, le système commence par faire une marche aléatoire dans le réseau de confiance de l'utilisateur. Si le produit est trouvé, la marche aléatoire se termine et la cote est attribuée comme étant une estimation de la cote réelle au produit. Sinon, selon une probabilité qui varie en fonction de la profondeur, la marche continue. Comme la précision diminue plus que la recherche s'éloigne de la source, les chercheurs ont fixé la profondeur à 6. Ce chiffre vient de l'étude «The small world experiment» qui explique que partout à travers le monde, les gens sont séparés seulement par 6 personnes. Lorsque la marche s'arrête, c'est-à-dire qu'il n'y avait pas d'individu qui avait coté ce produit ou que la probabilité d'arrêt a été atteinte, le système prend la cote d'un produit similaire en calculant la corrélation de Pearson.

O'Donovan et al.[36] propose un système qui utilise un réseau au filtrage collaboratif. L'idée principale est lorsqu'un utilisateur veut un produit, il peut demander conseil à son entourage. Toutefois, pour un produit différent l'utilisateur va peut-être demander à un entourage différent. Ou encore, il se peut que les personnes à qui il demande conseil pour un produit n'aient pas de préférence similaire pour un tout autre produit, par exemple de la musique et des films. Afin de déterminer l'entourage d'un utilisateur, le système vérifie si la cote prédite par chacun des membres de son entourage pour un produit est semblable à la cote fournie par l'utilisateur. Ensuite, il détermine le degré de fiabilité des membres de l'entourage selon le nombre de recommandations bien prédites par rapport au nombre total de recommandations.

1.1.3 Règles d'association

Les règles d'association sont généralement utilisées afin de découvrir des motifs qui se répètent dans des ensembles de données qui peuvent être de très grandes dimensions. Tout comme les réseaux de confiance, les règles d'association sont souvent jumelées avec une approche par filtrage collaboratif afin de rendre ce dernier encore plus précis. De manière générale, les règles d'association suivent un modèle probabiliste de cause à effet. C'est-à-dire que le système analyse une multitude d'évènements et par la suite, regroupe les instances qui précèdent un fait. Par exemple, si dans un super marché la plupart des gens achètent dans une même visite des couches, de la purée pour bébé et lors d'un futur achat du lait en poudre, alors le système pourrait créer une règle qui indiquerait que s'il y a des couches et de la purée dans un même panier, il est fort probable que l'utilisateur achètera du lait en poudre.

Les systèmes de recommandations basés sur les règles d'association utilisent généralement un historique de transactions afin d'extraire les règles qui permettront aux systèmes de

faire les associations [32]. Cependant, ces méthodes conduisent souvent à un très grand nombre de règles, ce qui rend les connaissances obtenues à partir de ces règles inefficaces [31]. Qui plus est, les méthodes de règles d'association exigent des paramètres explicites tels que le nombre de règles et le niveau de confiance associé à ces règles [32].

Dans des travaux antérieurs [28], un système de recommandations a été créé en se basant sur des règles d'association, qui sont produites grâce à l'utilisation d'un modèle qui effectue un contrôle sur la qualité de celles-ci dès la création. À la base, il y a deux types d'association : les associations entre les produits et les associations entre les utilisateurs. Par exemple,une association entre les produits peut être que les utilisateurs qui aiment le produit A et le produit B aiment aussi le produit C. Pour ce qui est de l'association entre les utilisateurs, ceci pourrait être que les articles qui sont aimés par l'utilisateur A et l'utilisateur B sont aussi aimés par l'utilisateur C. Cette pratique permet de trouver des inférences entre les règles et le fait d'analyser le chevauchement entre les utilisateurs permet d'émettre des recommandations aux utilisateurs qui n'ont pas beaucoup de similarité directement entre eux.

Bien que l'utilisation d'un modèle pour créer des règles d'association augmente la qualité des règles d'association, le problème n'en demeure pas moins que le nombre de règles produites reste élevé. Par conséquent, le bruit causé par un très grand nombre de règles réduit la confiance attribuée aux recommandations de façon significative [31].

1.1.4 Modèle latent

Contrairement aux approches qui utilisent un réseau de confiance ou un filtrage collaboratif, les systèmes de recommandations qui utilisent des modèles latents ne vont pas garder les informations des utilisateurs en mémoire lors de la recommandation. Ils vont plutôt utiliser ces informations afin de créer des modèles en utilisant des techniques de

forage de données. Afin de créer un modèle, le système commence par entrainer son modèle avec des paramètres aléatoires et ensuite, raffine les paramètres avec une ou plusieurs fonctions de coût. Contrairement à l'approche en mémoire, l'approche des modèles latents ne va pas nécessairement tenter de trouver des liens directs entre les utilisateurs et les produits. Ils vont plutôt tenter de découvrir des liens latents qui vont motiver l'achat d'un produit par l'utilisateur. Suivant cette idée, il est alors possible de capturer des faits qui pourraient motiver certains utilisateurs à préférer certains produits sans que cela soit évident pour un système traditionnel. Toutefois, il est difficile avec cette approche d'expliquer le raisonnement derrière une recommandation, car les lignes directrices n'ont pas d'étiquettes proprement identifiées. C'est-à-dire qu'un facteur latent peut être une mixture de différents facteurs réguliers.

L'élaboration d'un modèle latent dans le domaine des systèmes de recommandations emploie souvent la technique de factorisation de matrice. Cette technique permet d'associer les utilisateurs et les produits dans un nouvel espace où les liens entre les utilisateurs et les produits peuvent être définis par l'utilisation du produit scalaire. La matrice de l'espace initiale est généralement formée d'utilisateurs pour les lignes et de produits pour les colonnes. Ainsi, pour calculer une cote estimée pour un produit, il faut prendre le vecteur de l'utilisateur et celui du produit dans l'espace latent et ensuite, calculer le produit scalaire entre ces deux vecteurs. Toutefois, comme les ensembles de données pour les systèmes de recommandations sont souvent dispersés, il est important de faire attention pour ne pas ajouter trop de bruit dans le nouvel espace [27].

Un autre outil qui est très populaire lors de l'élaboration de modèle latent est le processus décisionnel de Markov. L'idée générale du processus décisionnel de Marcov est la suivante : le système démarre à l'intérieur d'un graphe à un état initial s_0 et peut choisir n'importe quelle transition selon une probabilité P possible à partir de s_0. Shani et al.[47] redéfinit la problématique des systèmes de recommandations comme étant un problème

d'optimisation de séquençage de processus plutôt qu'un système de prédiction standard. Afin de résoudre cette nouvelle définition, ils utilisent les chaînes de Markov. Ils définissent les états comme étant les historiques d'achats des utilisateurs et limitent l'ordre des chaînes à k comme étant un chiffre entre 1 et 5 afin de ne pas avoir un espace d'états trop grand. Pour ce qui est des transitions, ils modèlisent la probabilité qu'un utilisateur achète un produit x' s'il a acheté $x_1, x_2, ..., x_k$ au préalable. Toutefois, la faiblesse de ce modèle relève du fait que l'historique observé de l'utilisateur est très court.

Dans [23], les auteurs créent un système de recommandations basé sur un modèle latent pour résoudre le problème de l'utilisation d'informations implicites. Ils appliquent leur méthode à un domaine de télévision où ils recommandent des chaînes. Dans leur méthode, ils déterminent le nombre de fois qu'une émission de télévision spécifique est regardée par un utilisateur. Puis, en utilisant ce nombre comme une partition avec un poids, ils créent une matrice de préférences. Même si leur méthode présente de bons résultats, plusieurs paramètres doivent être estimés et trouver les valeurs optimales peut prendre beaucoup de temps. De plus, dans le domaine des historiques d'achat général, un utilisateur va généralement acheter un produit qu'une seule fois.

1.2 Validation

La validation des systèmes de recommandations propose différents défis [20]. Lorsque les chercheurs développent un algorithme, ils utilisent souvent un ensemble de données bien précis. Par exemple, un système qui recommanderait des livres selon le résumé de celui-ci ne pourrait pas être validé de la même façon qu'un système qui recommande des films selon les cotes fournies par les utilisateurs. Aussi, le but d'un système de recommandations n'est pas le même pour tout le monde. Certains veulent offrir une expérience nouvelle à leurs utilisateurs [37] alors que d'autres [47] veulent maximiser la marge de profit en

recommandant des produits dont il est possible d'acheter plusieurs accessoires complémentaires. Même s'il devenait possible de valider un système en retirant toutes traces d'erreur, il a été démontré [20] que les utilisateurs ne donnent pas les mêmes cotes selon la période à laquelle ils sont sollicités. Ainsi, différentes techniques permettent d'évaluer les différents systèmes de recommandations selon leurs spécifications particulières.

Dans un monde idéal, il serait possible de demander aux utilisateurs si les produits proposés sont satisfaisants. Toutefois, comme cela demande beaucoup de temps et de ressources, l'utilisation de validation théorique est préférable. À cette fin, différents ensembles de données sont disponibles afin de valider les systèmes de recommandations selon différentes caractéristiques. Par exemple, *MovieLens* [19] offre des ensembles de données où un très grand nombre d'utilisateurs ont émis des cotes en fonction de leurs préférences pour des films. *Jester* [16] est un ensemble de données qui contient plusieurs cotes pour des blagues où les défis sont au sujet de la compréhension sémantique. *Epinion* [35] est un ensemble de données qui lui, en plus de contenir un ensemble d'utilisateurs et des cotes pour différents produits, contient de l'information en ce qui concerne les relations de confiance entre les utilisateurs.

Une autre possibilité pour valider le système de recommandations est l'utilisation d'un ensemble de données synthétiques [20]. L'avantage d'un tel ensemble de données est la possibilité de simuler des scénarios probables et de soumettre le système à d'autres types de tests. Par exemple, avec un ensemble de données synthétiques, il est possible de tester comment le système se comporte lorsqu'il y a beaucoup de nouveaux utilisateurs, lorsqu'un utilisateur a des préférences particulières ou encore, lorsque le nombre d'utilisateurs devient très grand. Toutefois, il faut faire attention aux résultats des tests, car les données simulées ne reflètent pas toujours la réalité. Ainsi, bien que cela soit peut être pratique pour tester rapidement un nouvel algorithme, il est quand même important de valider le système avec un ensemble de données réelles.

1.2.1 La racine de l'erreur quadratique moyenne

La racine de l'erreur quadratique moyenne «Root Mean Square Error» est une mesure qui permet de calculer la différence entre une cote réelle (r) et une cote prédite ($\widehat{r}$) pour un ensemble de recommandations n. Cette mesure est utilisée surtout pour les systèmes qui utilisent des cotes comme information explicite. D'autres variantes de cette mesure sont la moyenne d'erreur absolue et la moyenne normalisée d'erreur absolue.

$$RootMeanSquareError = \sqrt{\frac{\sum_{i=1}^{n}(r_i - \widehat{r}_i)^2}{n}} \tag{1.4}$$

Selon l'objectif du système, il peut devenir intéressant de regarder seulement l'erreur quadratique moyenne, car cette mesure met beaucoup l'emphase sur l'écart entre la cote prédite et la cote réelle. Ainsi, dans un environnement où on s'intéresse surtout à la différence entre les cotes, cette mesure est essentielle [20]. Toutefois, pour un système où l'écart n'a que très peu d'importance, par exemple un système binaire, d'autres mesures seraient plus adéquates. Dans ce même ordre d'idée, il est intéressant d'utiliser la moyenne normalisée d'erreur absolue lorsqu'il est nécessaire de comparer les résultats obtenus d'un système avec des systèmes où les cotes ne suivent pas la même échelle de gradation.

1.2.2 Précision, rappel et mesure-f

La précision et le rappel sont les deux mesures les plus populaires pour évaluer un système de recherche d'informations [20]. Afin d'utiliser ces mesures, il faut définir ce qu'est de l'information pertinente, qui sont de bonnes recommandations dans le domaine des systèmes de recommandations, et de l'information non pertinente, qui sont de mauvaises recommandations.

La précision est définie comme étant le nombre de bonnes recommandations par rapport

aux nombres totaux de recommandations soumises à l'utilisateur.

$$Précision = \frac{Nb.\ de\ bonnes\ recommandations}{Nb.\ total\ de\ recommandations} \tag{1.5}$$

Le rappel est définie comme étant le nombre de bonnes recommandations par rapport aux nombre total de bons produits parmi l'ensemble de données.

$$Rappel = \frac{Nb.\ de\ bonnes\ recommandations}{Nb.\ total\ de\ bons\ produits} \tag{1.6}$$

Bien que ces mesures soient très populaires, lorsqu'elles sont utilisées directement, leurs résultats peuvent être biaisés par le nombre d'éléments utilisé. C'est pourquoi une autre mesure, qui est la moyenne harmonique des deux dernières, est plus utile. Cette nouvelle mesure, appelée la F-Mesure [20], fait un équilibre entre la précision et le rappel. Par exemple, un système pourrait très bien recommander un très petit nombre de produits et ainsi obtenir une précision très haute. Dans ce même ordre d'idée, un système n'aurait qu'à recommander l'ensemble des produits pour obtenir un rappel parfait. Or, en utilisant la F-Mesure, le système doit faire un compromis entre la précision de ses recommandations et le nombre de produits couvert.

$$F - mesure = \frac{2 * Precision * Rappel}{Precision + Rappel} \tag{1.7}$$

Bien qu'en théorie la mesure du rappel semble bien fonctionner, ce n'est pas le cas en pratique. Afin de mettre efficacement cette mesure en pratique, il faudrait demander à une grande partie des utilisateurs d'utiliser/d'acheter/de visionner tous les produits afin de déterminer quels sont les bons produits. Toutefois, il est possible d'estimer le nombre de bons produits en prenant ceux pour lesquels il y a une majorité de cotes et ensuite faire la moyenne, mais cela est biaisé envers les produits populaires. Herlocker et al.[20]

propose de faire la recommandation des N meilleurs produits pour lesquels une cote existe. Pour ce faire, ils utilisent une partie des produits cotés par l'utilisateur pour entrainer un modèle et ensuite, utilisent la partie restante pour créer les recommandations. En faisant cela, ils assument que la distribution des bons et moins bons produits pour un utilisateur est semblable à la distribution globale des produits. Toutefois, cela limite les recommandations lorsqu'un utilisateur a un petit historique d'achats.

1.2.3 Courbe ROC

La courbe ROC est un outil de mesure permettant de mesurer la différence entre la prédiction d'une donnée pertinente et de bruits. Tout comme la précision et le rappel, cette mesure nécessite qu'une distinction se fasse entre une bonne recommandation et une mauvaise recommandation. Le résultat de cette mesure fournit un graphique tel qu'illustré par la figure 1.1 sur laquelle il est possible de voir la performance du système. Dans ce graphique, l'axe des y représente le pourcentage de bonnes recommandations et l'axe des x représente le pourcentage de mauvaises recommandations.

Ainsi, en regardant la courbe, il est possible de voir si le système est performant. Sachant qu'un système aléatoire obtiendrait une diagonale, appelé ligne de non-discrimination, il est important pour le système d'être au dessus de celle-ci. Afin de comparer le résultat pour différents systèmes, au lieu d'utiliser directement les courbes et de les comparer, il est plus simple d'utiliser l'aire sous la courbe comme mesure de performance, car celle-ci est indépendante quant au nombre de recommandations effectuées.

En plus d'être une bonne mesure de validation, la courbe ROC permet de visualiser les forces et faiblesses du système [20]. En traçant la courbe, il devient plus facile de voir le compromis à faire entre la précision et la couverture voulue. Pour ce faire, il s'agit de regarder le point où le nombre de recommandations proposées affecte significativement

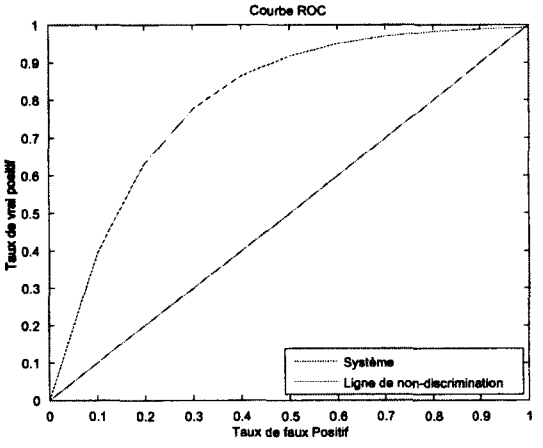

Figure 1.1 – Courbe ROC

le pourcentage de mauvaises recommandations.

1.3 Défis généraux

Lors de la mise en place d'un système de recommandations, qu'il utilise de l'information implicite ou explicite, celui-ci doit faire face à certains problèmes pratiques s'il veut demeurer efficace dans son utilisation de tous les jours.

1.3.1 Démarrage à froid

Lorsque le système de recommandations rencontre un nouvel élément, que ce soit un produit ou un utilisateur, il doit le gérer comme étant un cas spécial. Tel que mentionné dans [20], le problème du démarrage à froid survient lorsqu'il n'y a pas assez d'informa-

tion pour être en mesure d'émettre des recommandations. Par exemple, lorsqu'un nouvel utilisateur s'ajoute au système, il n'a pas encore d'historique. Évidemment, il est impensable d'offrir des recommandations personnalisées à un utilisateur qui n'est pas, ou très peu, connu du système.

Afin de surmonter ce problème, le système peut décider de ne pas émettre de recommandations à un utilisateur dont le système n'a pas atteint un certain nombre d'informations requises, car il n'est pas capable de cibler ses préférences avec le peu d'information disponible. Toutefois, cette pratique limite l'efficacité du système et est à éviter. Une solution plus bénéfique à la fois pour l'utilisateur et pour le système de recommandations est d'utiliser d'autres sources d'informations. En ayant ainsi un système hybride, par exemple un système utilisant à la fois le filtrage collaboratif et le filtrage par contenu [45], il est possible d'aider l'utilisateur à faire un choix judicieux en lui offrant des recommandations, peu importe s'il est nouveau ou non. Les systèmes vont utiliser de l'information basée sur le contenu jusqu'à temps que l'usager ou le produit atteignent un certain nombre d'activités, des achats dans le cas d'informations implicites ou un nombre satisfaisant de cotes dans le cas d'informations explicites, il y a une transition qui se fait de l'approche par contenu vers une approche par collaboration.

Un autre problème similaire au démarrage à froid pour les produits est le «long tail problem» [38]. La problématique s'explique comme étant une file où les produits les plus populaires se retrouvent à la tête de la file et que les produits les moins populaires, ou encore les nouveaux produits dans le système, se retrouvent dans la queue. Comme les systèmes de recommandations comportent un nombre important de produits et qu'une grande partie de ces produits se retrouvent dans la queue, il est primordial de trouver des solutions qui incluent ces produits. La solution proposée par [38] afin de surmonter ce problème est d'utiliser la segmentation des produits de la queue afin d'estimer les cotes manquantes. Après avoir trouvé le point de séparation entre la tête et la queue, [38] effec-

tue une segmentation des produits en utilisant un algorithme expectation-minimisation et crée des modèles de prédiction pour chacun des segments avec des séparateurs à vaste marge [8].

1.3.2 Données dispersées

Ce défi, similaire au démarrage à froid, se distingue du fait que le problème n'est pas tant le manque de données causé par un nouvel utilisateur ou un nouveau produit, que la diversité des produits et des préférences des utilisateurs [1]. Ce problème suit le même principe que le problème des grandes dimensionalités, la malédiction de la dimensionnalité. Par exemple dans le domaine de la recommandation de films, si un utilisateur a des préférences particulières, il ne sera pas évident pour un système utilisant un filtrage collaboratif de trouver d'autres utilisateurs avec des préférences similaires. Aussi, du côté des produits, si très peu d'utilisateurs cotent certains films, même avec de très bonnes cotes, ceux-ci ne seront recommandés que très rarement.

Comme dans le cas du démarrage à froid, des recherches ont montré qu'en combinant d'autres sources d'informations, par exemple des données démographiques [39], il était possible d'obtenir une meilleure mesure de similarité entre les individus. Une autre approche pour résoudre le problème des données dispersées est d'utiliser la décomposition en valeurs singulères. Sarwar et al.[42] utilise cette technique afin de réduire le nombre de dimensions et ainsi, être en mesure d'offrir une mesure de similarité qui rejoint un plus grand nombre d'utilisateurs.

1.3.3 Défis techniques

De nos jours, les plateformes utilisant les systèmes de recommandations contiennent un très grand nombre de produits et d'utilisateurs. Dans le cas des systèmes utilisant un filtrage collaboratif, ils utilisent un voisinage d'utilisateurs ou de produits afin d'émettre leurs recommandations. Toutefois, avec ces différents voisinages qui ne cessent d'accroître, il est nécessaire pour les systèmes de recommandations d'utiliser des techniques leur permettant de gérer le tout sans avoir de perte de performance. Aussi, lorsque l'utilisateur visite un site où des recommandations sont possibles, il est important que le système émette ses recommandations rapidement.

Une technique proposée afin de surmonter ce problème est l'utilisation de segmentation [44]. Ainsi, en créant des partitions d'utilisateurs, cela réduit l'ensemble d'utilisateurs à la taille de la partition. Lorsque le système va recommander ses produits à l'utilisateur, au lieu d'utiliser tous les utilisateurs pour trouver ses voisins, il va utiliser seulement les utilisateurs qui font partie du segment qui reflète au mieux les préférences de celui-ci. Ainsi, l'espace de recherche est plus petit et du même coup, le temps d'exécution se voit améliorer.

1.3.4 Tendance en fonction du temps

Dans le domaine des systèmes de recommandations, certains utilisateurs auront toujours le même profil. C'est-à-dire que les préférences de ces utilisateurs ne vont pas changer avec le temps [49]. Toutefois, il est possible pour les préférences d'un utilisateur de changer pour différentes raisons. Plus ce changement de comportement se fait tard dans la vie de l'utilisateur pour le système, plus il sera difficile d'émettre des recommandations qui reflètent ces nouvelles préférences. Toutefois, si l'utilisateur n'est pas très vieux dans le système, celui-ci est plus apte à s'adapter plus rapidement au changement de préférences

de l'utilisateur. Ce phénomène s'applique également pour les produits. Lorsqu'un produit devient très populaire, il va souvent se retrouver en haut de la liste des recommandations. Toutefois, avec le temps, un produit qui était très populaire à une époque peut ne plus convenir aux besoins présents des utilisateurs.

À cette fin, Xiang et al.[49] propose une solution qui utilise les préférences des utilisateurs à long terme et à court terme. Afin de modéliser les préférences des utilisateurs à court et long terme, ils utilisent un graphe pour représenter les relations entre les produits et les utilisateurs. La représentation des utilisateurs et des produits se fait à l'aide de tuple du type <Utilisateur, Produit> et <Session, Produit>. Ainsi, il est possible de faire des recommandations sur les préférences à long terme (en utilisant le tuple <Utilisateur, Produit>) ainsi que des recommandations avec les préférences à court terme (en utilisant le tuple <Session, Produit>.

1.4 Défis de l'information implicite

Même si le domaine des systèmes de recommandations gagne en popularité, les systèmes de recommandations qui reposent uniquement sur l'information implicite sont relativement peu utilisés par rapport à ceux qui emploient des informations explicites [51]. En plus des défis généraux rencontrés par les systèmes de recommandations, les systèmes utilisant seulement de l'information implicite doivent faire face à plusieurs difficultés. Comme ils utilisent une source d'informations complètement différentes des systèmes à information explicites, les solutions pour ceux-ci ne peuvent pas s'appliquer. Ces difficultés sont expliquées en détail dans [23]. Ici, nous les résumons pour faciliter la compréhension du mémoire. En fait, ces difficultés sont les principales raisons pour lesquelles nous ne pouvons pas simplement utiliser les techniques qui ont su prouver leur efficacité lors de l'utilisation d'information explicite telle que les cotes.

1.4.1 L'absence de commentaires des utilisateurs

Contrairement à la situation avec de l'information explicite, où l'utilisateur peut indiquer ses préférences vis-à-vis un produit, l'information implicite ne donne pas toujours de renseignements clairs sur les préférences de l'utilisateur. Entre autres, il n'est pas possible de connaître les raisons qui motivent l'utilisateur à ne pas acheter un produit. Il se pourrait que l'utilisateur ne soit pas intéressé par le produit, mais cela peut aussi signifier que le prix du produit est trop élevé, que le produit n'était pas disponible au moment d'effectuer ses achats, qu'il n'a simplement pas vu le produit, etc.

D'autre part, en observant le comportement d'un utilisateur en fonction de son historique d'achats, nous pouvons extraire des comportements répétés tels que les préférences à l'égard d'une catégorie spécifique de produits. En se basant sur ces comportements répétés, nous pouvons être confiants de l'intérêt d'un utilisateur face à ce genre de produits. À titre d'exemple, si un utilisateur loue plusieurs films d'action et quelques films d'horreur, on peut alors être confiant quant à ses préférences penchants vers les films d'action et ce, même si nous n'avons pas d'information explicite de sa part.

1.4.2 Cohérence des données

Lorsqu'il n'y a pas d'informations explicites, il est difficile, voire impossible, de faire la différence entre les achats effectués par l'utilisateur qui seront utilisés par celui-ci et les achats effectués à d'autres fins. Dans de nombreux cas, un utilisateur peut acheter un cadeau pour quelqu'un, louer un film pour un membre de la famille, etc. Cela crée beaucoup de mauvaises informations dans les données qui peuvent influencer négativement les recommandations si ce problème n'est pas pris en charge. Avec l'information explicite, ce problème a moins d'impact, car il est possible pour l'utilisateur d'indiquer ses préférences directement avec ce produit.

1.4.3 Confiance vs préférence

Dans le cas présent où on utilise des informations implicites, la connaissance que nous recueillons (par exemple, le nombre d'occurrences où des produits d'une catégorie spécifique ont été achetés) indique une confiance à l'égard des goûts de l'utilisateur pour une catégorie spécifique. Ainsi, nous devrions être plus intéressés par des motifs répétés à l'intérieur d'un historique d'achats qu'un achat unique. Un achat unique peut être fait pour des raisons différentes. En contre-partie, avec des informations explicites, les renseignements fournis par l'utilisateur indiquent une préférence pour un produit spécifique directement en fonction d'un niveau de satisfaction. Par conséquent, un seul achat peut donner des informations utiles sur les préférences de l'utilisateur pour ce produit.

1.4.4 Besoin de redéfinir les métriques d'évaluation

Avec les informations explicites, il existe déjà des métriques d'évaluations bien connues pour la validation des systèmes de recommandations basées sur la comparaison des prédictions entre la cote des produits recommandés par rapport à la cote réelle, comme l'erreur quadratique moyenne (RMSE) [20]. Toutefois, avec les données implicites, les raisons qui motivent un utilisateur à ne pas acheter un produit ne sont pas claires ; en fait, cela est aussi vrai pour un achat. Nous devons trouver des façons plus subtiles de découvrir des similitudes entre les utilisateurs. Avec l'absence de commentaire de la part de l'utilisateur, nous devons être en mesure de faire la différence entre les éléments pour lesquels l'intérêt des utilisateurs est élevé et ceux pour lesquels l'intérêt est faible.

1.5 Besoin d'une technique de segmentation

Les méthodes qui utilisent des informations explicites ont des techniques qui leur permettent de regrouper des utilisateurs similaires, comme les réseaux de confiances, les cotes, etc. Toutefois, avec les informations implicites ce choix est plus restreint. Des techniques comme la mesure du cosinus sont utilisées pour calculer la similitude entre deux utilisateurs, mais afin de créer des voisinages, il est important d'avoir une mesure permettant le calcul de la similitude entre un utilisateur et un ensemble d'utilisateurs.

Dans ce chapitre, nous avons vu comment créer un système de recommandations. Toutefois, le domaine des systèmes de recommandations qui utilisent des informations implicites nécessite encore beaucoup de recherche afin d'être compétitif avec les systèmes qui utilisent de l'information explicite. À cette fin, nous croyons que des méthodes de segmentations pourraient améliorer la précision des systèmes de recommandations en découvrant des similitudes entre les utilisateurs qui peuvent ne pas être apparentes avec des techniques traditionnelles.

Dans le prochain chapitre, une étude est faite sur différentes techniques de segmentation qui visent aussi bien des ensembles de données avec des attributs catégoriques que numériques. Cette étude a pour but de déterminer si une technique pourrait faciliter de façon considérable l'étape de la découverte des similarités entre les utilisateurs.

CHAPITRE 2

Segmentation

Lorsqu'il y a beaucoup d'informations inconnues dans un ensemble de données, la segmentation, un outil de forage de données, permet de trouver des relations entre ces données qui sont cachées. Contrairement à la classification, où il y a des éléments connus pour ce qui est des relations entre les données, la segmentation est utilisée lorsqu'il n'y a pas, ou très peu, d'information en ce qui concerne les classes des données. Le but premier d'un algorithme de segmentation est de regrouper les données similaires en segments de sorte que ces données aient une forte similitude à l'intérieur du segment et une forte dissimilitude lorsqu'on les compare aux autres segments. Ainsi, le résultat d'une segmentation génère des segments qui sont tous différents les uns des autres.

Le résultat d'une segmentation peut être complètement différent selon la technique utilisée. Certaines techniques se basent sur la densité des données comme DBSCAN, un algorithme populaire qui regroupe les données se retrouvant à l'intérieur d'un rayon donné. D'autres méthodes se basent sur une hiérarchie de segments où, selon l'approche, il est possible de trouver des sous-segments à l'intérieur même d'un segment. Lors du déroulement d'un algorithme de segmentation, il y a deux façons d'attribuer les données

aux segments. La première est l'attribution stricte. Cette attribution affecte les données à un seul segment. Ainsi, à la fin de la segmentation, toutes les données ont seulement une étiquette. La seconde est l'attribution floue. Ce type d'attribution permet à une donnée d'appartenir à différents segments selon un nombre entre 0 et 1[24].

Afin de trouver des motifs qui se répètent pour la création des segments, il est important de bien définir la mesure de similitude entre les données. Une des mesures très populaires, qui s'applique aux données numériques, est la mesure de distance Euclidienne. L'intuition de cette mesure est d'utiliser la distance physique entre les données. Cette mesure fonctionne bien lorsqu'il y a des segments isolés ou compacts, mais l'inconvénient primaire de cette mesure est que les attributs avec une plus grande échelle ont tendance à dominer les autres attributs [24]. Il existe une multitude de mesures selon le type de données qui est utilisé. Toutefois, il y a deux façons importantes de mesurer les données entre elles. La première est de mesurer les données deux à deux, par exemple lorsqu'il y a un point central à l'intérieur du segment et qu'il faut mesurer la similitude de ce point avec une nouvelle donnée. Cependant, il se peut que pour différentes données, une mesure deux à deux ne soit pas suffisante afin de déterminer si une donnée appartient à un segment et qu'il soit nécessaire de comparer cette donnée avec plusieurs autres données en même temps [2]. La seconde technique est de mesurer la similitude entre une donnée et plusieurs autres. Ce qui permet de trouver des caractéristiques additionnelles afin de regrouper les données en segments [50].

Afin de trouver différents segments dans un ensemble de données, il y a deux principales approches utilisées dans le domaine de la segmentation. La première est la segmentation en partition. Cette segmentation permet d'obtenir différents segments tous indépendants les uns des autres, alors que la seconde est la segmentation hiérarchique. Cette dernière modélise un arbre où il est possible d'avoir plusieurs niveaux de segmentation. Dans le cas de la segmentation hiéarchique, il est possible d'obtenir un résultat soit en appliquant

une approche par division, ou par agglomération. L'approche par division commence par regrouper les données dans un segment appelé la tête. Ensuite, par l'utilisation d'une mesure de similitude, les données sont divisées en segments et cela continue jusqu'à ce que la qualité de la segmentation ne puisse plus être améliorée. Contrairement à l'approche par division, l'approche agglomérative isole chacune des données comme étant un segment et regroupe ensuite les segments qui proposent une meilleure similitude ensemble. L'avantage d'utiliser une approche hiérarchique par rapport à une approche par partition est la possibilité de découvrir des sous-segments à l'intérieur d'un segment. Toutefois, afin de ne pas avoir trop de segments à la fin de l'algorithme, il faut généralement définir un critère d'arrêt comme un nombre maximal de segments.

Dans le domaine de la segmentation, il est souvent nécessaire de faire un compromis entre le nombre d'attributs à utiliser et le nombre de données disponibles. Le problème lié à un très grand nombre d'attributs, aussi appelé dimension, est que plus il y a d'attributs, plus il est possible pour les données d'être différentes. Par exemple, si on utilise des points sur un plan cartésien et que la segmentation se fait seulement sur l'axe des x, la segmentation sera simple et il est très probable qu'il y ait suffisamment de similitude entre les données pour avoir de bons segments distincts. Toutefois, si la segmentation se fait sur 3 dimensions, c'est-à-dire en ajoutant l'axe des y et des z, il est probable qu'il n'y ait pas assez de données pour faire la segmentation. Ainsi, comme le but d'une segmentation est de regrouper les données qui sont similaires entre elles, plus il y a d'attributs et plus le nombre de données requises afin d'avoir une segmentation efficace doit être grand. Ceci dit, les deux défis liés à la segmentation de données de très grandes dimensions sont les données dispersées, c'est-à-dire les données qui ne sont pas similaires à aucune autre donnée, et l'efficacité de la segmentation en présence de bruit dans l'ensemble de données, en terme d'exactitude et de vitesse d'exécution [21]. Dans le domaine du forage de données, cette problématique porte le nom de malédiction de la dimension. Afin de

surmonter cette problématique, des chercheurs [50],[21],[3],[4],[13] projettent les données dans un nouvel espace afin de réduire le nombre de dimensions tout en gardant une variance qui se rapproche de la variance initiale entre les données dans l'espace d'origine.

Dans un ensemble de données, il y a souvent des données qui ne se conforment pas aux motifs trouvés, segments, et il peut être nécessaire de trouver ces données afin d'analyser certaines particularités d'un système. Par exemple, les systèmes bancaires utilisent la détection d'anomalie afin de détecter les fraudes, un système médical peut détecter un signe de cancer chez un patient, ou encore un mauvais produit dans une chaîne de montage [11]. Dans le domaine de la segmentation, il y a deux types de données non-conformes qui nous intéresse soit : le bruit et les données aberrantes. Le bruit est une donnée qui peut avoir été perturbée lors de la cueillette d'information. Par exemple, le bruit dans un signal radio peut être causé par un autre produit électronique qui émet des ondes sur une fréquence similaire. En revanche, une donnée aberrante est une donnée qui n'a pas subi de distorsion, mais qui ne ressemble pas aux autres données de son ensemble. Lors de la segmentation d'un ensemble de données, il y a différentes approches [22] pour détecter les données aberrantes. Il est possible de faire une segmentation standard et d'établir une étiquette pour les données qui, basé sur un seuil donné, ne sont pas similaires aux segments. Il est également possible de créer des modèles dont un servira à segmenter les données dites *régulière* et l'autre servira à segmenter les données aberrantes.

2.1 Type d'attributs

Selon le domaine applicatif de la segmentation, le type d'attributs peut être différent. Ainsi, il y a principalement deux types d'attributs : les attributs numériques et catégoriques. La segmentation de données numériques se fait sur des ensembles de données où les attributs peuvent seulement contenir des valeurs numériques. Par exemple, la seg-

mentation des étudiants en fonction des résultats de leurs travaux et examens. Afin de ne pas avantager les attributs qui ont une plus grande échelle, une des premières étapes est de normaliser les données. Ensuite, afin de déterminer la similitude entre les données, différentes mesures peuvent être utilisées. La mesure Euclidienne, qui est un cas particulier de la mesure de Minkowski (eq. 2.1) lorsque $p = 2$, est une mesure très populaire et est souvent utilisée pour des données qui sont en 2 ou 3 dimensions [24].

$$Minkowski(x, y, p) = (\sum_{i=1}^{n} |x_i - y_i|^p)^{1/p} \tag{2.1}$$

Contrairement à la segmentation de données numériques, la segmentation de données catégorique ne peut pas utiliser des mesures de similitude comme Minkowski. Ceci est expliqué par le fait qu'il n'y a pas encore de façon universelle permettant de convertir des catégories en nombres. Toutefois, il existe différentes approches afin de créer des segments avec des données catégoriques. Une approche standard est d'exploser les catégories en attributs binaires, où chacune des options d'une catégorie représente un attribut. Par exemple, s'il existe une catégorie *Couleur des cheveux* et que cette catégorie contient 5 choix, après avoir explosé la catégorie *Couleur des cheveux* il y aurait 5 nouveaux attributs pouvant avoir la valeur 1 ou 0 et chacun des attributs corresponderaient à une couleur précise. Néanmoins, cela a pour effet d'augmenter grandement le nombre de dimensions et peut nuire à la qualité de la segmentation. Il est aussi possible d'utiliser une structure d'attributs représentée par un arbre, où le parent d'un noeud représente sa généralisation [24]. À titre d'exemple, un noeud parent pourrait être *«Meuble»* alors que ses enfants pourraient être des spécialités comme *«Lit»*, *«Sofa»*, *«Table»*, *etc.* Une mesure qui est souvent utilisée pour trouver la similitude entre des vecteurs de données binaires est la mesure de cosinus.

2.2 Techniques de segmentation avec données catégoriques

Les attributs catégoriques offrent des défis différents des attributs numériques. Avec les attributs catégoriques, il n'y a pas de mesures évidentes pour déterminer la similitude entre les catégories. Aussi, il est important de considérer le fait qu'il peut exister des sous-segments à l'intérieur d'un même segment. Par exemple, dans le cas où la segmentation se fait sur des utilisateurs où leurs historiques d'achats sont utilisés comme attributs, il est possible qu'il existe des sous-ensembles d'individus avec des préférences différentes face à certains produits à l'intérieur du même segment [50]. Il existe déjà un bon nombre d'algorithmes pour traiter des attributs catégoriques. Toutefois, les algorithmes existants ne sont pas encore aussi efficaces que les algorithmes qui traitent des attributs numériques. Les principaux inconvénients des techniques existantes sont expliqués par Xiong et al.[50], mais sont résumés ici afin de faciliter la compréhension.

Certains de ces algorithmes nécessitent la configuration de plusieurs paramètres, comme le nombre de segments attendu, ce qui peut être difficile à calibrer afin d'obtenir de bons résultats. De plus, la calibration des paramètres pour un ensemble de données peut donner des résultats complètements différents avec un autre ensemble. Ainsi, il est préférable d'utiliser un algorithme qui ne demande pas de paramètres afin de mettre tous les efforts sur la segmentation et non sur l'optimisation des paramètres.

Un autre problème qui affecte les algorithmes existants est la dépendance entre l'ordre de lecture des données et le résultat. Il n'est pas naturel pour un algorithme de segmentation de produire différents résultats selon l'ordre auquel les données ont été lues. Évidemment, un bon algorithme doit être en mesure de toujours produire le même résultat avec le même ensemble de données indépendamment de l'ordre de lecture.

Finalement, la complexité élevée d'un algorithme peut demander beaucoup de temps afin de produire un résultat. Afin de surmonter ce problème, les systèmes vont faire un échantillonnage des données et créer leurs premiers segments pour ensuite affecter les données aux segments. Toutefois, cette technique dépend de l'échantillon et n'est donc pas stable dans le cas où l'échantillon recueilli ne contient pas de segments bien définis. Bien que les techniques existantes offrent des résultats adéquats, il est nécessaire de trouver de nouvelles méthodes afin d'améliorer la précision et l'efficacité des algorithmes de segmentation d'attributs catégoriques tout en évitant les problèmes mentionnés ci-dessus.

2.2.1 ROCK

ROCK [17] est un algorithme de segmentation utilisant une approche hiérarchique agglomérative pour des attributs catégoriques. Cet algorithme utilise une notion de liens entre les données. Afin de créer les segments, l'algorithme commence par calculer les voisins qui ont une plus grande similitude en utilisant le coefficient de Jaccard, où A et B représentent des historiques de transactions.

$$Jaccard(A, B) = \frac{|A \bigcap B|}{|A \bigcup B|} \tag{2.2}$$

Après avoir calculé les voisins pour chacune des données, selon un seuil de similitude Θ, ils introduisent la notion de liens. Un lien est le nombre de voisins communs entre deux données. Ainsi, si le nombre de voisins est élevé, il est alors probable que ces deux données appartiennent au même segment. Afin de déterminer la qualité d'un segment, les auteurs utilisent une fonction d'optimisation par rapport au nombre de liens à l'intérieur d'un segment. Bien que l'idée d'utiliser des liens entre les données semble justifiée, la complexité élevée de l'algorithme causé par le calcul des voisins rend la segmentation sur

des ensembles de données très larges inefficace. Il est aussi important de préciser que le démarrage de l'algorithme se fait par échantillonnage, ce qui a pour effet de rendre la segmentation instable.

2.2.2 CACTUS

CACTUS [14] est un autre algorithme de segmentation pour des attributs catégoriques qui se déroule en trois phases. La première phase de cet algorithme résume l'ensemble de données en comptant les pairs d'attributs qui se répètent. Ensuite, selon un certain seuil fixé par un paramètre, l'algorithme conserve seulement les pairs qui sont fortement reliées. Lors de la seconde phase, la segmentation, détermine des segments potentiels sur les pairs d'attributs. Par la suite, ils augmentent le nombre d'attributs pour ainsi créer des triplets, quadruplets, et ainsi de suite. Lorsque l'algorithme obtient tous les segments potentiels, il commence la phase de la validation, qui consiste à créer les véritables segments en se basant sur le support des segments potentiels. Ce support est fixé selon un paramètre α qui peut être difficile à ajuster selon l'ensemble de données [50]. De plus, les segments sont créés en se basant sur l'espace initial des données. Toutefois, avec des données réelles qui contiennent des attributs catégoriques, les segments ont plus de chance de se former dans différents sous-espaces [50].

2.2.3 COOLCAT

Plutôt que d'utiliser des mesures comme le coefficient de Jaccard, certains chercheurs proposent une nouvelle approche qui utilise l'entropie afin d'obtenir une segmentation. COOLCAT [5] est un algorithme qui suit cette approche et qui est incrémental, c'est-à-dire qu'il est capable d'ajouter de nouvelles données sans avoir à recalculer tous ses segments. L'algorithme commence par effectuer un échantillonnage sur l'ensemble des

données et trouve les k données qui sont les plus différentes selon l'entropie. Ensuite, il se sert de ces k données comme étant les segments initiaux et affecte incrémentalement le reste des données aux segments ayant la plus petite entropie. Une des limites de cette approche est que le résultat de la segmentation dépend beaucoup de l'ordre selon laquelle les données ont été lues. Cette problématique réduit l'efficacité de l'algorithme, car selon un ensemble de données fixe, le résultat d'une segmentation devrait toujours être identique.

2.2.4 AT-DC

Une autre approche pour faire la segmentation d'attributs catégoriques est proposée par [10]. AT-DC, qui est un algorithme n'ayant besoin d'aucun paramètre, fonctionne selon l'hypothèse que des segments sont des groupes où un très grand nombre de données se croisent. Ainsi, en se basant sur cette hypothèse, un critère a été établi pour permettre à AT-DC de s'arrêter automatiquement lorsque la qualité des segments ne peut plus être améliorée. AT-DC utilise une approche hiérarchique de haut en bas. Il commence par regrouper l'ensemble des données dans la tête et ensuite, tente de séparer les données en deux segments de façon itérative. Si une nouvelle coupe améliore l'homogénéité des données, alors cette coupe est conservée et le noeud parent est retiré. AT-DC utilise l'index de *Gini* dans le but d'ordonner les données en ordre d'importances pour l'initialisation de la coupe. AT-DC contient également une phase d'optimisation qui consiste à déplacer une donnée à la fois vers un nouveau segment si la qualité de la segmentation peut être améliorée. Finalement, l'algorithme se termine lorsque le produit de la segmentation est stable. La stabilité est atteinte lorsque la séparation des segments ne peut plus améliorer la qualité de la segmentation. Le fait que cet algorithme ne demande aucun paramètre est très bien. Toutefois, l'ordre dans lequel les éléments sont évalués lors de la séparation initiale a un impact significatif sur la qualité de la segmentation [50].

2.3 DHCC

Afin d'offrir une solution qui ne demande pas de paramètres, qui est indépendant de l'ordre dans lequel les données sont lues, qui est capable de trouver des sous-segments à l'intérieur d'un segment et qui est performant avec de très larges ensembles de données, Xiong et al.[50] propose une approche en se basant sur une segmentation hiérarchique par division en étudiant la problématique de la segmentation d'attributs catégoriques comme étant un problème d'optimisation. DHCC est un algorithme qui produit un dendrogramme de segments pour des données ayant des attributs catégoriques et se base sur l'analyse des correspondances multiples afin d'initialiser ses segments.

En vue de travailler avec les attributs catégoriques, la première étape pour DHCC est de faire une transformation des catégories en attributs binaires. Ainsi, en explosant les catégories, il en découle une matrice indicatrice Z où chacune des lignes représentent une donnée et chacune des colonnes représentent un attribut provenant d'une catégorie. Par exemple, une catégorie qui contient 5 attributs aura seulement une de ses colonnes à 1 et le reste sera composé de 0. Ainsi, pour chaque ligne de Z, la sommation des attributs sera toujours le nombre de catégories provenant de l'ensemble de données initial. Lorsque la matrice indicatrice est créée, la seconde phase de l'algorithme est d'effectuer l'analyse des correspondances multiples afin d'obtenir les facteurs qui représentent au mieux les données dans un nouvel espace de plus petites dimensions. Un exemple de Z est présenté dans le tableau 2.1, où *Sexe*, *Taille* et *Occupation* représentent des catégories avec leurs attributs respectifs.

Cette seconde étape est nécessaire afin de trouver des structures à l'intérieur de données nominales. Lorsque cette technique est utilisée avec une matrice indicatrice, l'analyse produit une projection des données dans un nouvel espace qui contient moins d'attributs. Dans le but de réduire le nombre d'attributs, l'algorithme va calculer la décomposition en

Sexe		Taille		Occupation		
Homme	Femme	Petit	Grand	Professeur	Étudiant	Autre
1	0	1	0	1	0	0
0	1	0	1	0	1	0
1	0	1	0	0	0	1

Tableau 2.1 – Exemple d'une matrice indicatrice

valeur singulière de la matrice indicatrice, qui produit les axes principaux des données, et garder seulement les plus significatifs. Ainsi, il est possible d'utiliser des ensembles de données qui contiennent un très grand nombre d'attributs sans que l'efficacité de l'algorithme en soit affectée. Les similitudes entre les données sont découvertes en utilisant la mesure de Khi^2 entre les données transformées dans la matrice indicatrice Z et les données projetées dans le nouvel espace T par l'analyse.

Afin d'optimiser la segmentation, les auteurs utilisent la mesure du Khi^2(eq. 2.3) entre une donnée et un ensemble de données afin de définir la fonction objective. Ainsi, l'optimisation se fait sur la somme de l'erreur de Khi^2, où C_k représente un segment, z_i représente un objet quelconque provenant des données et d_{Chi}(eq. 2.4) la similitude entre une donnée et le centre d'un segment.

$$Erreur\ de\ Khi^2 = \sum_{k=1}^{K} \sum_{z_i \in C_k} d_{Chi}(z_i, C_k) \tag{2.3}$$

$$d_{Chi} = \sum_{j} \frac{(z_{ij} - \bar{c}_{kj})^2}{\bar{c}_{kj}} \tag{2.4}$$

Où z_{ij} est l'élément (i, j) de la matrice Z. Le centre d'un segment $\bar{c}_{kj}$(eq. 2.5) est obtenu en calculant la racine carrée des fréquences des catégories dans un segment C_k, où $z.j$ représente la somme de z_{ij} pour tous les z_i appartenant à C_k.

$$\bar{c}_{kj} = \left(\frac{z.j}{|C_k|}\right)^{1/2} \tag{2.5}$$

2.3.1 Fonctionnement de l'algorithme DHCC

L'algorithme de DHCC se résume comme suit. L'algorithme commence par faire un segment, la tête, et le sépare en deux. Ceci produit ainsi un arbre binaire de segment. Afin de séparer un segment, DHCC effectue un fractionnement préliminaire et ensuite, raffine ce fractionnement pour augmenter la qualité des segments. Le tout s'exécute jusqu'à ce que la qualité du segment ne puisse plus être améliorée.

Fractionnement préliminaire

Le fractionnement préliminaire se fait en utilisant le résultat de l'analyse des correspondances multiples sur l'ensemble de données du segment à fractionner. Comme l'analyse produit un nouvel espace où la première dimension est la plus significative, DHCC utilise cette première dimension afin de faire la première séparation des données. Par conséquent, comme la séparation est binaire, les données dont les coordonnées dans le nouvel espace sont négatives se retrouvent dans le noeud de gauche, et les donnes dont les coordonnées sont positives se retrouvent dans le noeud de droite. Le fait d'utiliser seulement la première dimension assure la rapidité de l'algorithme pour obtenir ce fractionnement préliminaire. Comme la première dimension est celle qui couvre le plus la variance des données, le fait d'utiliser cette dimension pour la séparation initiale des données permet en général de générer un fractionnement de bonne qualité.

Raffinement

Suite au fractionnement préliminaire, le raffinement est en charge de déplacer toutes les données à l'intérieur des nouveaux segments afin d'optimiser la qualité de la segmentation. Ainsi, pour chacune des données nouvellement segmentées, DHCC calcule la similitude entre la donnée et les deux segments. Pour effectuer ce calcul, il est nécessaire de retirer temporairement la donnée de son segment initial, car la mesure de Khi^2 nécessite que la donnée soit à l'intérieur du segment pour lequel la similitude est calculée. Ce raffinement s'effectue jusqu'à ce que les segments deviennent stables, c'est-à-dire que les données ne transigent plus d'un segment à l'autre.

Critère d'arrêt automatique

Comme un des avantages de DHCC est de ne pas avoir à fournir de paramètre, il est donc essentiel de trouver une approche automatique pour déterminer quand la segmentation doit être terminée. À cette fin, les auteurs proposent d'utiliser une mesure de qualité qui est globale à l'ensemble de la segmentation pour déterminer si un segment doit être fractionné. À première vue, il serait possible d'utiliser l'erreur de Khi^2 et d'optimiser cette fonction. Toutefois, comme DHCC diminue toujours l'erreur en fractionnant des segments, cela produirait un arbre où chacune des feuilles représenteraient une donnée. Donc, il est important de trouver une mesure d'efficacité qui fait un bon compromis entre la qualité de la segmentation et le nombre de segments finaux. Pour ce faire, les auteurs proposent une nouvelle mesure de qualité basée sur l'entropie où T représente l'ensemble de données.

$$Q(C_p) = \frac{|C_p|}{n}\Big(\sum_{j=1}^{J} p(v_j|C_p)\log p(v_j|C_p) - \sum_{j=1}^{J} p(v_j|T)\log p(v_j|T)\Big) \tag{2.6}$$

Cette mesure à elle seule n'est pas suffisante pour résoudre la problématique, car elle ne tient pas compte du nombre de segments finaux. Ainsi, tel que présenté dans l'équation suivante, ils incorporent le nombre de segments lors du calcul de la segmentation globale.

$$Q(C) = \sum_{C_i \in C} \frac{|C_i|}{n} Q(C_i) \tag{2.7}$$

Ainsi, DHCC calcule pour toutes les feuilles de l'arbre s'il est avantageux pour la segmentation de fractionner le segment. Le fractionnement est seulement avantageux si $Q(\{C_p\}) < Q(\{C_p^L, C_p^R\})$, où C_p^L et C_p^R représente les noeuds enfants de C_p. S'il y a fractionnement, alors l'algorithme continue en évaluant les nouveaux segments. Sinon, lorsque l'algorithme détecte que le fractionnement d'une feuille de l'arbre n'améliore pas la qualité de la segmentation, il étiquette ce noeud comme étant une feuille terminale et ne le revisite plus. Finalement, la segmentation se termine lorsque toutes les feuilles ont été étiquetées.

2.3.2 Avantages de DHCC

Un des avantages de DHCC est qu'il permet de ne pas avoir à préciser de paramètres lors de son utilisation, et ce de façon efficace. D'autres techniques existent comme par exemple la détection d'un «coude», où la qualité de la segmentation finale augmente de moins en moins par rapport aux nombres de segments produits. Ce genre d'approche est inefficace pour deux raisons. Premièrement, il est nécessaire de détecter le «coude», c'est-à-dire qu'il faut établir des mesures afin d'identifier lorsque celui-ci se produit. De plus, cette approche doit générer plus de segments qu'il est nécessaire en réalité pour détecter là où l'ajout de segments n'améliore plus de façon significative le résultat de la segmentation. Contrairement à DHCC, où l'algorithme se termine aux segments immédiats.

Un autre point important de DHCC est son indépendance à l'ordre selon lequel les données sont lues. Les raisons principales qui expliquent cette caractérisque sont les suivantes. Premièrement, comme il n'utilise pas d'échantillonnage pour la phase d'initialisation des segments et que l'analyse des correspondances s'effectue lorsque toutes les données sont lues, cela rend la première phase indépendante de l'ordre de lecture des données. Aussi, bien que la phase de raffinement travaille sur une donnée à la fois, elle met à jour les nouveaux segments seulement après avoir traité toutes les données. Ainsi, même si on répète la segmentation plusieurs fois sur le même ensemble de données et que les données sont lues dans un ordre complètement différent, le résultat sera toujours le même.

Comme DHCC utilise une approche hiérarchique, et que le résultat est un arbre binaire, il est en mesure de découvrir des sous-espaces à l'intérieur d'un segment. Par exemple, tel que présenté dans [50], lorsque l'algorithme est appliqué sur l'ensemble de données d'un Zoo, il commence par trouver deux segments dont l'un contient les mammifères. Ensuite, il est possible d'observer que l'algorithme associe des attributs qui sont plus dominants aux différents segments. Par exemple, un attribut très fort pour le segment mammifère est l'attribut binaire *fourrure* alors que l'attribut dominant de l'autre segment est *Ponds des oeufs.*

Afin de pouvoir être utilisable avec un très large ensemble de données, la complexité de DHCC est linéaire avec le nombre de données utilisées. L'ordre de DHCC est $O(\psi nJ \log K)$, où ψ représente la moyenne du nombre d'itérations de la phase de raffinement. Une grande partie du temps utilisée par l'algorithme est consacrée pour la phase de fractionnement préliminaire lors du calcul d'analyse des correspondances. La complexité d'une décomposition en valeur singulière, utilisée dans le cas d'une analyse des correspondances, pour une matrice nxM est $O(nMmin(n, M)))$. Il existe des techniques plus efficaces pour calculer la décomposition, qui donnent une complexité $O(nMr)$ lorsque $r < \sqrt{min(n, M)}$, où r signifie le rang de la matrice [7]. Une technique similaire utilise seulement les r

premières valeurs afin d'avoir une approximation de la matrice de données, ce qui rend le tout beaucoup plus rapide à calculer, mais affecte la précision du résultat.

2.3.3 Limites

Une des limitations de DHCC est l'espace occupé en mémoire par les données. Premièrement, lorsque la matrice indicatrice est créée, il est nécessaire d'exploser les catégories enfin de rendre chaque attribut binaire, ce qui a pour résultats d'augmenter de façon considérable le nombre d'attributs. De plus, lors de l'analyse des correspondances, les valeurs associées au calcul d'un nouveau segment sont de types *float.* ce qui fait en sorte que la mémoire utilisée pour la phase de fractionnement priliminaire de DHCC est de $|C_p| \times M$. Néanmoins, comme l'algorithme utilise une approche où la distance est calculée seulement avec le noyau du segment et non pas deux à deux avec tous les éléments, au lieu d'avoir une complexité quadratique, la complexité de DHCC est $O(nM)$.

Une autre difficulté éprouvée par DHCC est que les données aberrantes peuvent nuire au bon fonctionnement de l'algorithme. Lors de la phase de raffinement, lorsque l'algorithme transfère les données d'un segment à l'autre afin de déterminer quel segment offre la meilleure segmentation, la présence de données aberrantes affecterait le résultat. Une des solutions proposées serait de faire un pré-traitement sur l'ensemble de données initiales afin de retirer les données aberrantes.

2.4 Segmentation comme outil de système de recommandations

Les techniques de segmentation utilisées dans les systèmes de recommandations sont confrontés à plusieurs défis. Premièrement, les historiques d'achats sont considérés comme étant des ensembles de données avec des attributs catégoriques et il manque de mesure d'évaluation afin de déterminer la similarité entre les données. Le deuxième défi important est le grand nombre de produits que les magasins ont de nos jours, ce qui implique un très grand nombre de dimensions pour représenter les utilisateurs. Par conséquent, nous avons besoin d'une technique de segmentation qui peut traiter des données de très grandes dimensions. Troisièmement, comme nous voulons créer une méthode générique, la technique de segmentation doit avoir une façon automatique de déterminer le nombre de segments optimaux et ne doit pas nécessiter de paramètre. De plus, pour la problématique de recommandations que nous traitons dans ce travail, il n'est pas possible d'obtenir des informations comme les cotes ou toute autre variation de *score* pour les produits. La seule entrée est une matrice avec des attributs binaires qui indiquent l'achat ou non d'un produit par l'utilisateur.

DHCC est un algorithme de segmentation efficace pour le regroupement de données avec des attributs catégoriques qui a un moyen automatique de déterminer le nombre de segments nécessaires. Comme il est le premier algorithme de division hiérarchique pour la segmentation de données catégoriques, il a été conçu spécialement pour contrer la problématique des grandes dimensions et du manque de mesure de similarité efficace. Son application permet d'obtenir une bonne segmentation sur des utilisateurs qui peut être utilisée dans les systèmes de recommandations, où les clients dans un segment partagent des préférences similaires et les préférences d'un groupe peuvent être utilisées pour faire des recommandations aux utilisateurs à l'intérieur du segment. De plus, la struc-

ture hiérarchique de la segmentation basée sur le comportement d'achats de l'utilisateur permettrait d'être en accord avec la taxinomie des produits.

CHAPITRE 3

Méthodologie et Résultats

Dans le but de découvrir des informations cachées à propos des préférences parmi un ensemble d'utilisateurs, nous proposons d'exploiter la technique de segmentation hiérarchique par division. Nous avons comme hypothèse principale qu'un produit populaire a tendance à être un bon produit. Dans cet ordre d'idée, en regroupant les utilisateurs qui ont des goûts similaires, il est possible de recommander des produits ayant un très grand intérêt par ce groupe d'utilisateurs. En appliquant DHCC avec des informations implicites comme un historique d'achats, l'algorithme est en mesure de découvrir des groupes d'utilisateurs similaires qui sont organisés en forme d'arbre binaire. À partir de cet arbre, nous extrayons les produits qui ont le plus de chance d'être intéressants pour les utilisateurs dans une liste Top-N [46], et ce pour chacun des segments. La technique proposée offre des recommandations personnallisées pour chacun des utilisateurs dans chacun des segments.

Utilisateur	Produit a		Produit b	
	Acheté	Non Acheté	Acheté	Non Acheté
Utilisateur A	1	0	0	1
Utilisateur B	0	1	1	0
Utilisateur C	1	0	0	1

Tableau 3.1 – Exemple d'une matrice d'entrée

3.1 Segmentation

Afin d'utiliser DHCC, nous créons une matrice qui contient toutes les informations à propos des historiques d'achat pour chacun des utilisateurs. En traitant chaque produit comme étant une catégorie, nous créons une matrice avec les dimensions *Utilisateurs* X *Produits*, et explosons chacune des catégories afin de représenter les produits achetés et non achetés. Ceci transforme la représentation des données pour les transactions en vecteurs et cela est nécessaire pour le fonctionnement de DHCC avec notre ensemble de données. Le Tableau 3.1 montre un exemple de cette représentation dans le cas où il y a deux produits.

Dans DHCC, la similitude entre les utilisateurs (représentée par son historique d'achats) et un segment est calculée avec la mesure de Khi^2, c.-à-d.,

$$d_{Chi}(u_i, C_k) = \sum_j \frac{(\mu_{ij} - \bar{c}_{kj})^2}{\bar{c}_{kj}} \tag{3.1}$$

où μ_i est le vecteur des utilisateurs u_i de la matrice d'entrée et $\bar{c}_k$ est le vecteur qui représente le centre du segment C_k, qui est défini comme étant la racine carrée des fréquences des données catégoriques à l'intérieur du segment, ce qui devient la structure comportementale d'un segment C_k.

Tel que présenté dans la Figure 3.1, où chaque lettre représente un utilisateur et chaque feuille représente un segment, lorsque la segmentation est terminée, il en résulte une struc-

ture d'arbre binaire où chacune des feuilles représente un segment (dans le cas présent, il y a 5 segments). La raison principale pour laquelle nous nous intéressons davantage aux noeuds terminaux par rapport aux autres types de noeuds est qu'ils contiennent le plus d'information mettant en relief la différence entre les groupes d'utilisateurs (segments).

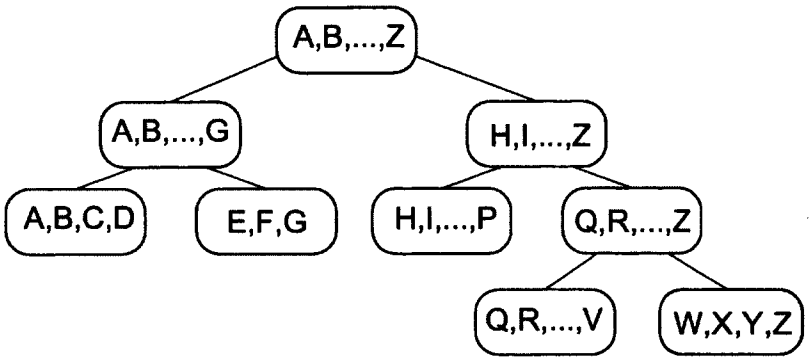

Figure 3.1 – Résultat d'une segmentation par DHCC.

Dans la section suivante, nous définissons les symboles qui seront utilisés pour le reste de la méthodologie (résumé dans le Tableau 3.2). P_c est la liste des produits populaires construite à partir des achats des utilisateurs au sein du segment c, $P_{c,n}$ est la liste des produits populaires, mais contient seulement les n premiers éléments et U_c est l'ensemble des utilisateurs pour le segment c.

3.2 Recommendations

Afin de créer d'intéressantes recommandations, nous avons d'abord supposé que plus un produit est populaire, plus il a tendance à avoir une bonne cote. Nous basons cette hypothèse sur le fait que les produits populaires peuvent avoir été promus par des sources externes comme des amis [25], la publicité, etc. Basés sur cette hypothèse, les produits

Symbole	Description
n	Nombre d'élément à garder parmi la liste des produits populaires
C	Ensemble de segments
C_k	Segment spécifique k
c_k	Centre du segment C_k
U_c	Ensemble des utilisateurs dans le segment c
S_k	Ensemble de k sous-partitions d'utilisateur
R_u	Ensemble de recommandations pour l'utilisateur u
B_u	Ensemble de produits achetés par l'utilisateur u
P_c	Liste de produits populaires pour le segment c
$P_{c,n}$	Liste de n produits populaires pour le segment c
$r_{u,i}$	Cote de l'utilisateur u pour le produit i
$\widehat{r_{u,i}}$	Cote estimé de l'utilisateur u pour le produit i
$s_{u,i}$	Vaut $r_{u,i}$ s'il existe, $\widehat{r_{u,i}}$ sinon
t	Seuil afin qu'une recommendation soit considérée bonne
$G_{u,i,t}$	Vaut 1 si la recommandation i est considérée bonne pour l'utilisateur u selon t, 0 sinon
N_r	Nombre de recommandations dont la cote est $\geq$ à r

Tableau 3.2 – Résumé des notations

sont ordonnés selon leur nombre d'achats, en ordre décroissant, pour chaque segment C_k. Ensuite, pour chacun des utilisateurs $u \in U_c$, la liste de recommandations R_u est créée. Cette liste de recommandations est basée sur les produits les plus populaires P_c à partir du segment assigné à l'utilisateur par DHCC. Évidemment, les produits recommandés à un utilisateur n'ont pas été achetés par celui-ci au préalable comme définis dans l'Équation 3.2. Comme P_c est déjà ordonné en ordre décroissant, c'est-à-dire que le produit le plus populaire se retrouve en premier, il s'agit alors de prendre les n premiers éléments de P_c qui ne sont pas dans B_u afin de créer la liste Top-N [46].

$$R_u = P_c - B_u \tag{3.2}$$

Parce que nous utilisons une technique hybride impliquant la liste Top-N et une technique de voisinage, nous éliminons une partie du problème de cohérence des données décrit dans la problématique à la page 24. Ainsi, même si un utilisateur effectue un achat pour quelqu'un d'autre, il va quand même être assigné à un voisinage avec des préférences similaires et ainsi, obtenir des recommandations d'un grand intérêt pour lui. Évidemment, il est nécessaire que la majorité des achats effectués par l'utilisateur aient été faits selon ses propres choix afin d'obtenir des recommandations justes selon ses préférences.

La structure comportementale de chacun des segments découverts joue un rôle très important dans l'efficacité du système de recommandations. Des utilisateurs sont regroupés dans un segment significatif si leurs historiques d'achat forment une structure comportementale similaire, c.-à-d., les produits populaires préférés par ce segment d'utilisateurs ne sont pas nécessairement populaires pour tous les utilisateurs, mais seulement pour les utilisateurs de ce segment. Dans le même ordre d'idées, les produits qui sont populaires pour l'ensemble des utilisateurs peuvent ne pas être populaires pour quelques segments d'utilisateurs. Bien que l'historique d'achats pour chacun des utilisateurs ne couvre qu'une

partie de la structure comportementale découverte, il est possible de faire des recommandations pour chacun des utilisateurs qui reflètent, aux meilleurs des connaissances du système, les préférences du segment.

3.3 Résultats Expérimentaux

Alors que la plupart des systèmes de recommandations utilisent des informations explicites, nous avons obtenu de bons résultats, en nous basant sur notre hypothèse, avec seulement des informations implicites. Afin de tester notre hypothèse ainsi que notre système de recommandations, nous avons utilisé les données de MovieLens qui proviennent de GroupLens Research. Ces données sont très connues dans le domaine des systèmes de recommandations et sont souvent utilisées pour faire des bancs d'essai. Cet ensemble de données contient 100 000 cotes allant d'un à cinq, cinq étant la meilleure cote et un étant la moins bonne. Les données contiennent 943 utilisateurs et les cotes sont appliquées à 1682 films, avec au moins 20 cotes par utilisateur. Il est à noter que les cotes ont été utilisées uniquement pour valider notre méthode et non pour entrainer le système de recommandations. Afin de créer la matrice d'entrée pour DHCC, chacune des cotes est considérée comme une transaction puis la valeur de sa caractéristique (film) est fixée à 1. Il en résulte une matrice indicatrice de 943 lignes par 3364 colonnes, où 3364 est obtenu en dédoublant le nombre de films afin d'être utilisé dans la matrice indicatrice.

3.3.1 Validation de l'Hypothèse

Afin de valider notre hypothèse qu'un produit populaire a tendance à être un bon produit, nous définissons et appliquons un critère d'acceptation en utilisant les informations fournies par les cotes. Ceci est équivalent à comparer les résultats de notre système de

recommandations, qui n'utilise pas les cotes (information explicite), aux résultats d'un système utilisant le filtrage collaboratif qui utilisent les cotes. Un résultat convaincant serait qu'un pourcentage élevé de nos recommandations coïncide avec les recommandations effectuées par le système de filtrage collaboratif avec des cotes élevées (par exemple 4 ou 5).

En prenant les n produits les plus populaires provenant de P_c, nous examinons leurs cotes associées. Précisément, lorsqu'un utilisateur a donné une cote pour le produit, nous utilisons simplement cette cote. Toutefois, si l'utilisateur n'a pas donné de cote pour le produit, nous utilisons une cote estimée par la combinaison des cotes de son segment pour ce produit. Alors pour chacun des produits, la moyenne des cotes fournies par les utilisateurs d'un segment est calculée, comme nous pouvons le voir dans l'Équation 3.3 où u' a déjà émit une cote pour i, et ensuite ce processus est répété pour chacun des couples (u, i) pour lesquels nous n'avons pas de cote, et ce, pour tous les segments. Sous l'hypothèse où u appartient au segment c, la valeur de la recommandation pour u de l'item i est la moyenne des cotes donné par les utilisateurs à l'item i (eq. 3.3). Nous avons décidé d'associer la cote moyenne à un produit qui n'a pas été acheté, car suite à la segmentation, il arrive que certains segments contiennent un petit nombre d'utilisateurs et que le nombre de produits en commun à l'intérieur du segment soit par conséquent petit. Or, nous prenons la moyenne des cotes des utilisateurs qui contribuent au plus à la structure comportementale du segment pour définir l'appréciation collaborative du produit.

$$\widehat{r_{u,i}} = \sum_{u' \in U'_c} \frac{r_{u',i}}{|U'_c|} \tag{3.3}$$

Ces cotes, initialement disponible ou estimée, nous permettent de définir le critère d'acceptation en se basant sur les recommandations qu'un système avec le filtrage collaboratif

aurait fait en fonction d'un certain seuil t. L'Équation 3.5 ci-dessous décrit si un produit i serait recommandé à l'utilisateur u, conformément au seuil pour les cotes t.

$$s_{u,i} = \begin{cases} r_{u,i} & \text{Si existe,} \\ \widehat{r_{u,i}} & \text{Sinon.} \end{cases} \tag{3.4}$$

$$G_{u,i,t} = \begin{cases} 1 & \text{Si } s_{u,i} \geq t, \\ 0 & \text{Si } s_{u,i} < t. \end{cases} \tag{3.5}$$

Afin de comparer les recommandations populaires produites par les deux systèmes, nous utilisons les résultats de l'Équation 3.5 pour définir un score pour les recommandations produites par notre système. À partir de l'Équation 3.6 ci-dessous, nous obtenons le score de $P_{c,n}$ en prenant le rapport de la somme de $G_{u,i,t}$ sur n et $|U_c|$. Comme l'Équation 3.6 est la moyenne des recommandations qu'un système avec un filtrage collaboratif ferait à chacun des utilisateurs de U_c à partir des produits $P_{c,n}$, elle fournit un niveau de confiance associé à la qualité de $P_{c,n}$.

$$ratioH(P_{c,n}, t) = \frac{\sum_{i \in P_{c,n}} \sum_{u \in U_c} G_{u,i,t}}{n * |U_c|} \tag{3.6}$$

Maintenant, nous pouvons définir un score global pour valider notre hypothèse. Pour ce faire, nous procédons à une moyenne pondérée sur le rapport pour chacun des segments basés sur le nombre de ses membres, afin d'obtenir le taux global de notre méthode, tel qu'indiqué dans l'Équation 3.7.

$$GlobalRatioH = \frac{\sum_{c \in C} ratioH(P_{c,n}) * |U_c|}{\sum_{c \in C} |U_c|} \tag{3.7}$$

Le tableau 3.3 présente nos résultats en utilisant l'ensemble de données de MovieLens, obtenu en spécifiant le seuil des cotes t aux différentes valeurs possibles des cotes et en

variant n. D'après ces résultats, nous pouvons observer qu'il y a un grand pourcentage des n produits de la liste des produits populaires qui sont considérés comme étant de bons produits lorsque $t = 3$, pour la majorité des utilisateurs. De plus, il est important de constater que les résultats lorsque $t = 4$ sont encore très bons. Comme nous utilisons l'ensemble de données de MovieLens et que les cotes varient entre un et cinq, nous considérons la cote centrale (une cote de 3 pour cet ensemble de données) comme étant moyenne, alors qu'une cote supérieure au centre (une cote de quatre ou cinq) comme étant très bonne. Ainsi, les résultats dans le tableau 3.3 démontrent qu'en utilisant seulement des données implicites, environs 75% de nos recommandations dans les top 10 ou 67% de nos recommandations dans les top 50 corrobore avec les préférences des utilisateurs obtenues par des informations explicites/filtrage collaboratif. De plus, lorsque le seuil est sous la valeur centrale, seulement 6% des recommandations ne sont pas considérées comme étant bonnes.

	$n = 10$	$n = 25$	$n = 50$
$t = 2$	98.1548%	98.2821%	98.4687%
$t = 3$	93.9873%	94.0318%	94.6257%
$t = 4$	**74.9417%**	**70.3033%**	**66.7805%**

Tableau 3.3 – Ratio global des utilisateurs ayant une cote définie par le seuil t

3.3.2 Validation des recommandations

Afin de rappeler comment les recommandations sont produites, nous commençons par créer une liste de produits populaires pour chacun des segments trouvés et ensuite, le système recommande des produits qui sont les plus probables d'être d'un grand intérêt pour les utilisateurs. Il est important de se rappeler que les segments sont les noeuds feuilles de l'arbre binaire produit par DHCC. Afin de valider le système de recommandations, nous utilisons k-fold cross-validation [26] basée sur les utilisateurs afin de créer

k systèmes de recommandations. Ensuite, les différents systèmes sont confrontés à des utilisateurs dont ils n'ont aucun historique d'achat. Similaire à la méthode utilisée pour valider notre hypothèse, nous utilisons l'historique des cotes des utilisateurs comme banc d'essai. Nous comparons ensuite les recommandations du système avec les produits préalablement achetés par l'utilisateur et si une recommandation fait partie de son historique d'achat, nous utilisons sa cote. Si le produit n'avait pas été acheté par l'utilisateur, nous utilisons le filtrage collaboratif afin d'obtenir la cote combinée de son segment.

Tel que mentionné dans le chapitre 1, nous ne pouvons pas utiliser les techniques existantes de validation qui fonctionnent bien avec des données dont l'information est explicite. Avec l'information explicite, nous aurions employé un cas spécial de cross-validation, la technique leave-one-out [9]. Ainsi, pour chacun des utilisateurs, nous n'aurions qu'à enlever la cote attribuée pour un certain produit et ensuite, il aurait suffi de comparer la cote générée par le système de recommandations avec la cote attribuée par l'utilisateur. Toutefois, ce type de validation ne peut être utilisé avec des recommandations basées sur la popularité. Lorsque les produits recommandés se retrouvent à l'intérieur d'un segment, les recommandations sont effectuées en tenant compte de l'historique des transactions globales de ce segment. Ceci étant dit, nous ne voulons pas pénaliser un produit populaire même s'il ne se retrouve pas dans l'historique d'achat d'un utilisateur spécifique. De plus, ce n'est pas parce qu'une recommandation ne se retrouve pas dans l'historique d'achat de l'utilisateur que celle-ci n'est pas bonne.

Comme méthode d'évaluation pour les recommandations, nous avons décidé d'utiliser la précision [20]. La précision N_{ri}/N_{ari} se définit comme étant le nombre d'informations pertinentes (N_{ri}) qui se retrouvent parmi toutes les informations recueillies (N_{ari}). Dans le domaine des systèmes de recommandations, la précision se traduit comme étant le nombre de bonnes recommandations par rapport au nombre total de recommandations. Le nombre de bonnes recommandations est obtenu en utilisant $G_{u,i,t}$. Afin d'obtenir une

précision globale, tous les ratios sont ensuite moyennés pour obtenir la précision des recommandations.

$$Precision = \frac{\sum_{u \in U_c} \sum_{i \in P_{c,n}} G_{u,i,t}}{n * |U_c|} \tag{3.8}$$

Le rappel N_{ip}/N_{ipt} se définit comme étant le nombre d'informations pertinentes recueillies N_{ip} par rapport au nombre total d'informations pertinentes N_{ipt}. Dans le domaine des systèmes de recommandations, le rappel se traduit comme étant le nombre de bonnes recommandations faites par le système de recommandations implicite par rapport au nombre total de bonnes recommandations qu'il aurait été possible de faire en utilisant les cotes N_t. Malheureusement, il n'est pas possible d'utiliser le rappel pour valider notre approche, car nous limitons le nombre de recommandations à N. Il aurait été possible de l'utiliser si nous avions eu un ordre à l'intérieur des produits. Comme nous considérons le même score pour les produits qui ont la même cotè, il n'est pas possible d'ordonner les meilleurs produits, car ils ont tous une cote de 5.

Du même coup, nous ne pouvons pas utiliser la F-mesure, qui est une mesure populaire. Ceci s'explique parce que la F-mesure est en fait une moyenne harmonique du rappel et de la précision afin de ne pas privilégier davantage l'un par rapport à l'autre.

Afin de valider le système de recommandations, nous avons d'abord séparé les utilisateurs en k sous-ensembles S_k et ensuite entrainé les k systèmes de recommandations en utilisant $k-1$ sous-ensembles S_{k-1} des utilisateurs. L'ensemble restant S_t est utilisé afin d'évaluer les recommandations. Évidemment, chacun des systèmes de recommandations est entrainé et testé avec un ensemble différent d'utilisateurs. À titre d'exemple, si nous choisissons $k = 3$, le premier système de recommandations serait entrainé en utilisant le premier et le deuxième ensemble alors que le troisième ensemble servirait à tester le système. De même, le deuxième système de recommandations serait entrainé en utilisant

le deuxième et le troisième ensemble alors que le premier ensemble servirait à tester le système. Lorsque tous les k systèmes de recommandations sont entrainés, nous utilisons leur ensemble d'utilisateurs qui n'a pas été utilisé lors de l'entrainement afin d'évaluer les recommandations.

Pour chacun des utilisateurs dans S_t, le segment qui représente au mieux les préférences de l'utilisateur lui est assigné. Ensuite, pour chaque recommandation offerte à l'utilisateur, nous regardons dans l'historique d'achat de celui-ci afin de voir si la recommandation s'y retrouve. Si l'utilisateur a acheté précédemment la recommandation, nous utilisons sa cote directement. Sinon, nous utilisons la cote estimée de son segment pour cette recommandation obtenue par filtrage collaboratif. Finalement, nous calculons la précision afin d'obtenir le ratio des recommandations pour lesquelles les utilisateurs portent un grand intérêt.

On peut se demander s'il est nécessaire de conserver les produits qui sont déjà dans l'historique d'achats des utilisateurs. Toutefois, il est important de se rappeler que ce n'est pas parce qu'un produit est dans un historique d'achats que c'est nécessairement un bon produit. De plus, il faut garder en tête que les segments créés par DHCC ont été découvert avant l'assignation des utilisateurs de S_t. Dans un même ordre d'idée, les P_c ne tiennent pas compte des historiques d'achats des utilisateurs de S_t. Le Tableau 3.4 représente la précision des recommandations pour un différent nombre de recommandations pour différentes valeurs de t.

	$m = 5$	$m = 10$	$m = 20$
$t = 2$	97.1064%	97.766%	97.8351%
$t = 3$	92.2766%	94.0532%	94.1702%
$t = 4$	**63.2553%**	**74.4255%**	**69.75%**

Tableau 3.4 – Pourcentage des utilisateurs dont les cotes pour les recommandations sont définies par le seuil t

À partir de ces résultats, nous pouvons voir que l'ensemble des recommandations four-

nies aux utilisateurs, qui n'étaient pas dans les ensembles d'entrainements, comprend un nombre élevé de recommandations avec de très bonnes cotes. Lorsque le système fait face à de nouveaux utilisateurs, lorsque $m = 10$, 74% des recommandations sont meilleures que la cote centrale. La principale différence entre le tableau 3.4 et le tableau 3.3, c'est que dans le tableau 3.4, les utilisateurs provenant de l'ensemble de tests ne sont pas inclus lors que la création de $P_{c,n}$. Aussi, il est important de noter que le rendement est comparable à celui où l'ensemble de données en entier est utilisé. Par conséquent, le système propose un grand pourcentage de recommandations avec des cotes élevées.

3.3.3 Comparaison entre divers systèmes de recommandations

À titre de comparaison avec les résultats de notre système de recommandations, nous avons utilisé la plateforme MyMediaLite [15]. MyMediaLite nous permet d'utiliser plusieurs systèmes de recommandations qui ne nécessitent pas d'informations explicites. Les algorithmes utilisés pour comparer nos résultats sont les suivants : Top-N Globale, kNN pondéré (sur une base utilisateur et sur une base de produits), WR-MF [23] et BPR-MF [41].

La technique Top-N Globale est un système qui recommande les N produits ayant les plus hauts scores. Contrairement à la méthode que nous proposons, l'utilisation de Top-N Globale se fait sur l'ensemble des utilisateurs. Ainsi, cette technique n'offre pas de recommandations personnalisées et recommande les mêmes produits à l'ensemble des utilisateurs.

En contre-partie, kNN est une technique de filtrage collaboratif qui vise à trouver les k plus proches voisins (nearest neighbors) pour chacun des objets. Une fonction de similitude est nécessaire afin de déterminer la distance entre les éléments. Après avoir trouvé le voisinage le plus près de l'objet, un vote majoritaire se fait pour effectuer les recom-

mandations. kNN peut être basé autant sur les utilisateurs que les produits et utilise une fonction de similitude selon le type choisi. La mesure de similitude utilisée par [15] est la mesure cosine.

Dans un autre ordre d'idée, WR-MF est une technique basée sur les modèles à facteurs latents en utilisant la factorisation de matrice. La principale idée est d'associer un niveau de confiance pour les produits figurants dans l'historique d'achat des utilisateurs. Avec ce niveau de confiance, ils cherchent les facteurs latents pour les utilisateurs et les produits. Lorsqu'ils trouvent ces facteurs, ils utilisent une fonction de coût afin d'optimiser la confiance reliée à la recommandation d'un produit à un utilisateur. Bien que cette méthode offre de bons résultats, elle est très lourde en terme de calcul [23].

Pour ce qui est de BPR-MF, c'est une technique utilisant la factorisation de matrice avec une analyse bayésienne. Au lieu d'utiliser seulement la corrélation positive, ils utilisent également la corrélation négative en utilisant des triplets du type (usager, produit i, produit j) où le produit i est préféré par rapport au produit j. Ainsi, il est possible d'établir des rangs entre les produits. Toutefois, l'hypothèse sur laquelle ils se basent pour affirmer leur corrélation négative n'est pas convaincante. Selon eux, un produit qui n'est pas présent dans l'historique d'achat représente un produit qui a moins d'intérêt qu'un produit présent. Toutefois, avec l'information implicite, nous n'avons aucune information sur les motivations derrière ces choix tel que discuté dans le chapitre 1.

Afin d'évaluer les recommandations, nous avons utilisé l'ensemble de données MovieLens. Au lieu de comparer les produits recommandés directement entre eux, nous comparons les cotes, existantes dans les données initiales ou estimées, qui sont associées aux recommandations. De cette façon, si deux produits sont différents, mais ont la même cote, ils ont tous deux la même valeur de recommandations. Pour comparer efficacement la cote, nous utilisons une technique similaire à celle utilisée pour valider notre méthode en 3.2.2. En raison de certaines limitations de MyMediaLite, tous les modèles ont été entrainés

	Top-N Global	kNN pondéré utilisateur	kNN pondéré produit	WR-MF	BPR-MF	Notre système
$t = 2$	97.9215%	97.9852%	98.176%	**98.2185%**	98.1654%	98.1548%
$t = 3$	93.4889%	93.0117%	93.9767%	93.3404%	93.5419%	**94.0085%**
$t = 4$	66.8293%	74.2842%	72.7572%	73.5737%	72.6405%	**74.7508%**

Tableau 3.5 – Précision des recommandations définies par le seuil t pour des systèmes de recommandations utilisant des informations implicites, avec $m = 10$.

avec l'ensemble de données en entier. Cela veut dire que cette étude comparative se limite à l'efficacité d'apprentissage.

Pour calculer le score d'une recommandation de chacun des modèles entrainés, lorsque la recommandation pour l'utilisateur est déjà dans son historique d'achat, nous prenons sa cote directement. Toutefois, lorsque la recommandation est pour un produit qui n'est pas dans son historique d'achat, nous utilisons la cote moyenne de tous les utilisateurs qui ont le produit dans leur historique d'achat. On peut voir dans le Tableau 3.8 les scores des différents systèmes de recommandations avec un nombre m de recommandations établie à 10 pour différentes valeurs possibles de t.

Comme nous pouvons le voir, notre méthode est capable de recommander des produits de qualité supérieure par rapport à la méthode naïve Top-N globale. Bien que le résultat de cette méthode est comparable aux algorithmes qui offrent des recommandations personnalisées, on voit que lorsque le seuil est plus élevé, la technique perd de son efficacité. C'est pourquoi la pluspart des systèmes offrent des recommandations personnalisées. De plus, notre méthode surpasse également les systèmes de recommandations en comparaison lorsque la valeur du seuil est élevée. En d'autres termes, les recommandations de notre méthode sont plus susceptibles de correspondre aux produits dont les cotes associées sont de 4 ou 5. Alors que notre système de recommandations ne demande aucun paramètre additionnel, il est en mesure de se comparer avec des systèmes plus complexes et qui

nécessitent plusieurs paramètres (WR-MF, BPR-MF). Nous remarquons que l'approche de kNN pondéré avec les utilisateurs atteint également une performance similaire à la nôtre. Toutefois, il est important de noter que kNN exige que le nombre de voisins soit prédéterminé.

Finalement, il faut être conscient de la limite de la méthode de comparaison utilisée ici. En absence de rétroaction de la part de l'utilisateur, nous avons utilisé les cotes existantes ou estimées. Évidemment, cette méthode de comparaison ne peut être utilisée que sur des données avec informations explicites. La validation des méthodes de recommandations basées sur des informations implicites est encore une question ouverte.

CONCLUSION ET PERSPECTIVES

Conclusion

Le domaine des systèmes de recommandations, qui utilisent seulement des informations implicites, est encore très récent et il n'existe pas de mesure bien définie pour les évaluer. L'importance des recommandations, toujours avec des informations implicites, s'étend aux recommandations utilisant des informations explicites, car selon [51], seulement une très faible proportion des produits sont habituellement évalués. Actuellement, les systèmes de recommandations qui utilisent une combinaison d'informations explicites et d'informations implicites sont très peu nombreux. Pour exploiter les informations implicites, il est nécessaire de développer des techniques efficaces pour découvrir des groupes d'utilisateurs similaires et pour exploiter les similitudes entre ces utilisateurs afin d'améliorer la qualité des recommandations. D'autre part, nous devons faire face aux défis relatifs à la validation de ce genre de systèmes, car même si nous pouvons être confiants des recommandations offertes à un utilisateur, il n'y a aucune garantie en ce qui concerne son appréciation.

Les travaux présentés dans ce mémoire offrent une technique qui permet d'émettre des recommandations pour un ensemble d'utilisateurs en combinant des informations recueillies par une segmentation hiérarchique des utilisateurs et de leurs historiques d'achats. Cette

technique propose une nouvelle voie d'exploration dans le domaine des systèmes de recommandations qui n'utilisent pas d'informations explicites. Le modèle obtenu suite à l'exécution de l'algorithme regroupe des utilisateurs ayant un profil d'achat similaire et fait l'étoffe des produits populaires à l'intérieur d'un groupe afin d'offrir des recommandations personnalisées pour l'ensemble des utilisateurs. Les résultats expérimentaux illustrent que notre méthode permet de recommander des produits qui sont appréciés par les utilisateurs dans une proportion très élevée. Cette proportion se compare avantageusement aux principales méthodes existantes qui se basent sur l'information implicite. Par rapport à ces méthodes qui nécessitent que les valeurs d'un ou plusieurs paramètres soient fournies ou optimisées, notre méthode ne demande pas de paramètre.

Perspectives

Maintenant que nous sommes en mesure de proposer des recommandations qui ont le potentiel d'être d'un grand intérêt en utilisant uniquement des données implicites, il y a certains aspects de la méthode actuelle qui pourraient être revisés dans le but d'améliorer le travail présent.

Suite à l'application de DHCC, la structure obtenue fournit beaucoup d'informations qui n'est pas encore exploitées dans notre méthode. Comme DHCC utilise une représentation par arborescence binaire des données, il est possible étudier la dépendance entre chaque paire de noeuds menant à un segment. Une fois que les dépendances entre les noeuds seront comprises, nous pourrions fournir des recommandations fondées sur les noeuds qui sont similaires, mais pas identiques, aux préférences de l'utilisateur [37]. De cette façon, nous pourrions fournir un plus grand nombre de recommandations variées à l'utilisateur.

Aussi, avec l'état actuel de notre méthode, les segments doivent être reconstruits afin

de découvrir des sous-structures qui auraient pu apparaître par l'ajout de nouveaux utilisateurs. Bien qu'il soit possible d'ajouter un nouvel utilisateur au modèle existant, il se pourrait que l'ajout de plusieurs nouveaux utilisateurs forme une spécialisation à l'intérieur du segment qui pourrait ainsi être séparée en deux segments dont l'homogénéité serait plus forte. Dans une version future de cet algorithme, il faudrait avoir la capacité de mettre à jour la structure des segments sans avoir à tout reconstruire.

Bibliographie

[1] G. ADOMAVICIUS et A. TUZHILIN : 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*, 17:734–749, juin 2005.

[2] S. AGARWAL et J LIM : Beyond pairwise clustering. *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 2:838–845, 2005.

[3] CC AGGARWAL, J HAN, J WANG et PS YU : A framework for projected clustering of high dimensional data streams. *Proceedings of the Thirtieth international conference on Very large data bases*, 30:852–863, 2004.

[4] CC AGGARWAL, JL WOLF et PS YU : Fast algorithms for projected clustering. Dans *Proceedings of the 1999 ACM SIGMOD international conference on Management of data*, pages 61–72, New York, New York, USA, 1999. ACM Press.

[5] D BARBARÁ, Y LI et J COUTO : Coolcat : an entropy-based algorithm for categorical clustering. Dans *Proceedings of the eleventh international conference on Information and knowledge management*, CIKM '02, pages 582–589, New York, NY, USA, 2002. ACM.

[6] Daniel BILLSUS et MJ J PAZZANI : User modeling for adaptive news access. *User modeling and user-adapted interaction*, 10(2):147–180, 2000.

[7] Matthew BRAND : Fast low-rank modifications of the thin singular value decomposition. *Linear Algebra and its Applications*, 415(1):20–30, mai 2006.

[8] CJC BURGES : A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 43:377–385, 1998.

[9] Robin BURKE : Evaluating the Dynamic Properties of Recommendation Algorithms. Dans *Proceedings of the fourth ACM conference on Recommender systems*, pages 225–228, New York, New York, USA, 2010. ACM.

[10] Eugenio CESARIO, Giuseppe MANCO et Riccardo ORTALE : Top-down parameter-free clustering of high-dimensional categorical data. *IEEE Trans. on Knowl. and Data Eng.*, 19(12):1607–1624, décembre 2007.

[11] Varun CHANDOLA, Arindam BANERJEE et Vipin KUMAR : Anomaly detection. *ACM Computing Surveys*, 41(3):1–58, juillet 2009.

[12] Christian DESROSIERS et George KARYPIS : A comprehensive survey of neighborhood-based recommendation methods. *Recommender Systems Handbook*, pages 107–144, 2011.

[13] E ELHAMIFAR et R VIDAL : Sparse subspace clustering. Dans *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2790–2797, 2009.

[14] Venkatesh GANTI, Johannes GEHRKE et Raghu RAMAKRISHNAN : Dans *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '99, pages 73–83, New York, New York, USA, 1999. ACM.

[15] Z GANTNER, Steffen RENDLE, Christoph FREUDENTHALER et L SCHMIDT-THIEME : MyMediaLite : a free recommender system library. Dans *Proceedings of the fifth ACM conference on Recommender systems*, pages 305–308. ACM, 2011.

[16] Ken GOLDBERG, Theresa ROEDER, Dhruv GUPTA et Chris PERKINS : Eigentaste : A constant time collaborative filtering algorithm. *Information Retrieval*, (August), 2001.

[17] Sudipto GUHA, R RASTOGI et K SHIM : ROCK : A robust clustering algorithm for categorical attributes. *Information systems*, pages 1–25, 2000.

[18] Ido GUY, Naama ZWERDLING, Inbal RONEN, David CARMEL et Erel UZIEL : Social media recommendation based on people and tags. Dans *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '10, pages 194–201, New York, NY, USA, 2010. ACM.

[19] J HERLOCKER, JA A KONSTAN et J RIEDL : An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. *Information retrieval*, 5 (4):287–310, 2002.

[20] Jonathan L. HERLOCKER, Joseph a. KONSTAN, Loren G. TERVEEN et John T. RIEDL : Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53, janvier 2004.

[21] Alexander HINNEBURG et DA KEIM : Optimal grid-clustering : Towards breaking the curse of dimensionality in high-dimensional clustering. Dans *Proceedings of the 25th International Conference on Very Large Databases*, pages 506–517. Morgan Kaufmann, 1999.

[22] Victoria HODGE et Jim AUSTIN : A Survey of Outlier Detection Methodologies. *Artificial Intelligence Review*, 22(2):85–126, octobre 2004.

[23] Yifan HU, Yehuda KOREN et Chris VOLINSKY : Collaborative filtering for implicit feedback datasets. Dans *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 263–272. IEEE, 2008.

[24] AK JAIN, MN MURTY et PJ FLYNN : Data clustering : a review. *ACM computing surveys (CSUR)*, 31(3), 1999.

[25] Mohsen JAMALI et M ESTER : TrustWalker : a random walk model for combining trust-based and item-based recommendation. Dans *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 397–406. ACM, 2009.

[26] Ron KOHAVI : A study of cross-validation and bootstrap for accuracy estimation and model selection. Dans *Proceedings of the 14th international joint conference on Artificial intelligence*, vol. 14, pages 1137–1145. LAWRENCE ERLBAUM ASSOCIATES LTD, 1995.

[27] Yehuda KOREN, Robert BELL et Chris VOLINSKY : Matrix factorization techniques for recommender systems. *Computer*, pages 42–49, 2009.

[28] Jiye LI, Bin TANG et Nick CERCONE : Applying association rules for interesting recommendations using rule templates. *Advances in Knowledge Discovery and Data Mining*, pages 166–170, 2004.

[29] L LI, Dingding WANG, Tao LI, Daniel KNOX et Balaji PADMANABHAN : Scene : a scalable two-stage personalized news recommendation system. Dans *ACM Conference on Information Retrieval (SIGIR)*, pages 125–134, 2011.

[30] Ming LI, B DIAS, W EL-DEREDY et PJG J G LISBOA : A probabilistic model for item-based recommender systems. Dans *Proceedings of the 2007 ACM conference on Recommender systems*, pages 129–132. ACM, 2007.

[31] Yuefeng LI et Ning ZHONG : Ontology based web mining for information gathering. *Web Intelligence Meets Brain Informatics*, pages 406–427, 2007.

[32] Weiyang LIN, SA A ALVAREZ et C RUIZ : Efficient adaptive-support association rule mining for recommender systems. *Data Mining and Knowledge Discovery*, 6 (1):83–105, 2002.

[33] Fabián P. P. LOUSAME et Eduardo SÁNCHEZ : View-based recommender systems. Dans *Proceedings of the third ACM conference on Recommender systems*, RecSys '09, pages 389–392, New York, NY, USA, 2009. ACM.

[34] Stephen Paul MARSH : Formalising trust as a computational concept. Rapport technique, Department of Computing Science and Mathematics, University of Stirling, 1994.

[35] Paolo MASSA et Paolo AVESANI : Trust-aware recommender systems. Dans *RecSys '07 Proceedings of the 2007 ACM conference on Recommender systems*, vol. 20, pages 17–24, 2007.

[36] John O'DONOVAN et Barry SMYTH : Trust in recommender systems. Dans *Proceedings of the 10th international conference on Intelligent user interfaces*, IUI '05, pages 167–174, New York, NY, USA, 2005. ACM.

[37] Kensuke ONUMA, H TONG et Christos FALOUTSOS : TANGENT : a novel,'Surprise me', recommendation algorithm. Dans *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 657–666. ACM, 2009.

[38] YJ PARK et Alexander TUZHILIN : The long tail of recommender systems and how to leverage it. Dans *Proceedings of the 2008 ACM conference on Recommender systems RecSys 08*, page 11, New York, New York, USA, 2008. ACM Press.

[39] MJ PAZZANI : A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review*, (Lang 1995):393–408, 1999.

[40] Bruno PRADEL, Savaneary SEAN, J DELPORTE et S : A case study in a recommender system based on purchase data. *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 377–385, 2011.

[41] Steffen RENDLE, Christoph FREUDENTHALER, Zeno GANTNER et Lars SCHMIDT-THIEME : BPR : Bayesian personalized ranking from implicit feedback. Dans *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 452–461, Arlington, Virginia, United States, 2009. AUAI Press.

[42] B SARWAR, George KARYPIS, J KONSTAN et J RIEDL : Application of dimensionality reduction in recommender system-a case study. *Architecture (2000)*, 1625(1):264–8, 2000.

[43] Badrul SARWAR, George KARYPIS, Joseph KONSTAN, John RIEDL et J REIDL : Item-based collaborative filtering recommendation algorithms. Dans *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.

[44] BM SARWAR et George KARYPIS : Recommender systems for large-scale e-commerce : Scalable neighborhood formation using clustering. *Proceedings of the Fifth International Conference on Computer and Information Technology*, 50 (12):158–167, 2002.

[45] Andrew I. SCHEIN, Alexandrin POPESCUL, Lyle H. UNGAR et David M. PENNOCK : Methods and metrics for cold-start recommendations. *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '02*, 46(Sigir):253, 2002.

[46] Klaus SEYERLEHNER, Arthur FLEXER et Gerhard WIDMER : On the limitations of browsing top-N recommender systems. Dans *Proceedings of the third ACM conference on Recommender systems*, vol. 66, pages 321–324, New York, New York, USA, 2009. ACM.

[47] Guy SHANI, RI BRAFMAN et David HECKERMAN : An MDP-based recommender system. *The Journal of Machine Learning Research*, 6(2):1265–1295, 2005.

[48] P SYMEONIDIS, A NANOPOULOS, A N PAPADOPOULOS et Y MANOLOPOULOS : Collaborative recommender systems : Combining effectiveness and efficiency. *Expert Systems with Applications : An International Journal*, 34(4):2995–3013, mai 2008.

[49] Liang XIANG, Quan YUAN et Shiwan ZHAO : Temporal recommendation on graphs via long-and short-term preference fusion. *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, 29(2):723, 2010.

[50] Tengke XIONG, Shengrui WANG, André MAYERS et Ernest MONGA : DHCC : Divisive hierarchical clustering of categorical data. *Data Mining and Knowledge Discovery*, 24(1):103–135, mai 2011.

[51] Xujuan ZHOU, Yue XU, Yuefeng LI, Audun JOSANG et Clive COX : The state-of-the-art in personalized recommender systems for social networking. *Artificial Intelligence Review*, 37(C):1–14, mai 2011.